# AD/ADVANTAGE

MANTIS for Windows Language Reference Manual

P19-2302-01

CINCOM®
The Smart Choice®

# AD/Advantage® MANTIS for Windows Language Reference Manual

**Publication Number P19-2302-01**

Cincom Systems, Inc.
55 Merchant Street
Cincinnati, OH 45246-3732
U.S.A.

PHONE: (513) 612-2300
FAX: (513) 612-2000
WORLD WIDE WEB: http://www.cincom.com

**Attention:**

Some Cincom products, programs, or services referred to in this publication may not be available in all countries in which Cincom does business. Additionally, some Cincom products, programs, or services may not be available for all operating systems or all product releases. Contact your Cincom representative to be certain the items are available to you.

## Release information for this manual

*AD/Advantage MANTIS for Windows Language Reference Manual*, P19-2302-01, is dated November 30, 1998. This document supports Release 2.2.01 and higher of MANTIS for Windows.

### We welcome your comments

We encourage critiques concerning the technical content and organization of this manual. A Reader Comment Sheet is included at the end of the manual for your convenience.

*Cincom Technical Support for AD/Advantage*

| | |
|---|---|
| FAX: | (513) 612-2000<br>Attn: MANTIS Support |
| E-mail: | helpna@cincom.com |
| Phone: | 1-800-727-3525 |
| Mail: | Cincom Systems, Inc.<br>Attn: MANTIS Support<br>55 Merchant Street<br>Cincinnati, OH 45246-3732<br>U.S.A. |

# Contents

**About this book** xi

Using this document ... xi
    Document organization ... xi
    Conventions ... xii
MANTIS documentation series ... xv

**Introduction** 17

MANTIS for Windows ... 17
MANTIS for Windows considerations ... 19
Terminology used in this document ... 20
Data display modes ... 21
    Full-screen mode ... 22
    Scroll mode ... 25
    KILL and CTRL-BREAK ... 28
    Windowing ... 29
Keyboard operation ... 30
    MANTIS editing and windowing keys ... 32
    MANTIS action keys ... 35
    MANTIS macro keys ... 36
Signing on to MANTIS ... 37

**MANTIS language conventions** 39

MANTIS programming guidelines ... 39
Character set ... 40
Reserved words ... 42
Symbolic names ... 44
Variables and arrays ... 45
Language fundamentals ... 47
    Statements ... 47
    Commands ... 47
    Program lines ... 48
    Comments ... 49

Text considerations ........ 50
Storing text data ........ 50
Text substrings ........ 51
Text expressions ........ 53
Numeric considerations ........ 54
Storing numeric data ........ 54
Scientific notation ........ 54
When to use BIG and SMALL ........ 55
Arithmetic expressions ........ 58
Relational and logical expressions ........ 60
Indirect reference (&) ........ 63
Built-in functions ........ 64

**MANTIS programming language 65**

Statements and built-in functions ........ 65
ABS ........ 74
ACCESS ........ 75
ATN ........ 80
ATTRIBUTE ........ 81
ATTRIBUTE (PRINTER) ........ 87
BIG ........ 88
BREAK ........ 89
CALL ........ 90
CHAIN ........ 93
CHR ........ 95
CLEAR ........ 96
COMMIT ........ 98
CONVERSE ........ 100
COS ........ 107
CURSOR ........ 108
DATAFREE ........ 110
DATE (function) ........ 111
DATE (statement) ........ 112
DELETE ........ 114
MANTIS file delete ........ 114
External file delete ........ 116
DEQUEUE ........ 118
DO ........ 119
DOLEVEL ........ 122
E ........ 123
EDIT ........ 124
ENQUEUE ........ 125
ENTRY-EXIT ........ 126

EXP ........ 128
FALSE ........ 129
FILE ........ 130
FOR-END ........ 134
FORMAT ........ 136
FSI ........ 137
GET ........ 139
    MANTIS file GET ........ 139
    External file GET ........ 143
HEAD ........ 147
IF-ELSE-END ........ 149
INSERT ........ 151
    MANTIS file INSERT ........ 151
    External file INSERT ........ 153
INT ........ 155
INTERFACE ........ 156
KEY ........ 159
LANGUAGE ........ 160
LET ........ 161
LOG ........ 164
MODIFIED ........ 165
NEXT ........ 167
NOT ........ 168
NULL ........ 168
NUMERIC ........ 169
OBTAIN ........ 170
OUTPUT ........ 172
PAD ........ 174
PASSWORD ........ 177
PERFORM ........ 178
PI ........ 180
POINT ........ 181
PRINTER ........ 182
PRINTER= ........ 183
PROGFREE ........ 186
PROGRAM ........ 187
PROMPT ........ 189
RELEASE (statement) ........ 190
RELEASE (function) ........ 193
RESET ........ 194
RETURN ........ 195
RND ........ 196
SCREEN ........ 197
SCROLL ........ 199

SEED ........ 201
SET ........ 202
SGN ........ 205
SHOW ........ 206
SIN ........ 209
SIZE ........ 210
SLICE ........ 212
SLOT ........ 214
SMALL ........ 215
SQR ........ 216
STOP ........ 217
$SYMBOL ........ 218
TAN ........ 219
TERMINAL ........ 220
TERMSIZE ........ 221
TEXT ........ 222
TIME (function) ........ 224
TIME (statement) ........ 226
TRAP ........ 228
TRUE ........ 231
TXT ........ 232
UNPAD ........ 233
UNTIL-END ........ 236
UPDATE ........ 237
    MANTIS file UPDATE ........ 237
    External file UPDATE ........ 239
UPPERCASE ........ 241
USER ........ 242
USERWORDS ........ 243
VALUE ........ 244
WAIT ........ 245
WHEN-END ........ 246
WHILE-END ........ 248
ZERO ........ 250

**Programming techniques 251**

MANTIS Logical Terminal Interface ........ 251
    Building a map set in your program ........ 252
    CONVERSE ........ 255
    Multiple images of a single-screen design ........ 259
    Windowing ........ 260
    Clearing a map ........ 264
    Clearing a map set ........ 265

External DO 267
- Passing arguments 268
- Program architecture 269
- Internal DO vs. external DO vs. CHAIN 271
- Modularizing 275
- Automatic mapping 276
- Entity definition 278
- Frequency of DOs 278
- Debugging 278
- Program size 279
- Programming considerations 279
- Performance 280

Miscellaneous programming hints 281

**Compatibility considerations (personal computer and IBM mainframe) 283**

General considerations 285
- Size limits 285
- Key assignments 286
- Indirect reference 286
- Comments 286
- Kanji 286
- Screen design 287
- ASCII and EBCDIC collating sequences 288
- Field input 288
- Noninteractive input 288
- Mainframe to Windows 289

Programming statements and functions 290
- ACCESS statement 290
- ATTRIBUTE statement 290
- BREAK statement 291
- CHAIN statement 291
- CHR function 291
- CLEAR statement 291
- COMMIT statement 291
- CONVERSE statement 292
- DO statement 292
- EDIT statement 292
- FOR statement 292
- FSI function 293
- GET statement 293
- KEY function 293
- LANGUAGE function 293
- NEXT statement 293

NULL function ....................................................................................... 294
NUMERIC function.................................................................................. 294
OUTPUT statement ................................................................................ 294
PERFORM statement ............................................................................. 294
PRINTER= statement ............................................................................. 294
RELEASE statement............................................................................... 295
RETURN statement ................................................................................ 295
SCROLL statement................................................................................. 295
SET statement ....................................................................................... 295
SHOW statement.................................................................................... 295
SLICE and SLOT statements.................................................................. 296
$SYMBOL function................................................................................. 296
UPPERCASE function ............................................................................ 296

**Dissimilarity debugging** **297**

**Messages** **299**

**Index** **375**

# About this book

## Using this document

MANTIS® is an application development system that consists of design facilities (e.g., screens and files) and a programming language. This manual describes the MANTIS programming language.

### Document organization

The information in this manual is organized as follows:

**Chapter 1—Introduction**
Introduces the MANTIS application development workbench and its programming language.

**Chapter 2—MANTIS language conventions**
Describes the programming conventions used by MANTIS.

**Chapter 3—MANTIS programming language**
Describes the statements and built-in functions of the MANTIS programming language.

**Chapter 4—Programming techniques**
Provides suggestions for using some features of MANTIS.

**Chapter 5—Compatibility considerations (Windows and IBM mainframe)**
Discusses differences between MANTIS for Windows and MANTIS for the IBM mainframe, and special considerations for PC programs that are to migrate to the IBM mainframe.

**Appendix A—Dissimilarity debugging**
Provides debugging recommendations for dissimilarity errors.

**Appendix B—Messages**
Lists and explains all errors that may appear while using the MANTIS language or running a MANTIS program.

**Index**

## Conventions

The following table describes the conventions used in this document series:

| Convention | Description | Example |
|---|---|---|
| `Constant width type` | Represents screen images and segments of code. | `Screen Design Facility`<br>`GET NAME LAST`<br>`INSERT ADDRESS` |
| Slashed b (b̸) | Indicates a space (blank).<br><br>The example indicates that a password can have a trailing blank. | `WRITEPASSb̸` |
| Brackets [ ] | Indicate optional selection of parameters. (Do not attempt to enter brackets or to stack parameters.) Brackets indicate one of the following situations: | |
| | A single item enclosed by brackets indicates that the item is optional and can be omitted.<br><br>The example indicates that you can optionally enter a program name. | `COMPOSE [`*`program-name`*`]` |
| | Stacked items enclosed by brackets represent optional alternatives, one of which can be selected.<br><br>The example indicates that you can optionally enter NEXT, PRIOR, FIRST, or LAST. (NEXT is underlined to indicate that it is the default.) | $\begin{bmatrix} \underline{\textbf{NEXT}} \\ \textbf{PRIOR} \\ \textbf{FIRST} \\ \textbf{LAST} \end{bmatrix}$ |

| Convention | Description | Example |
|---|---|---|
| Braces { } | Indicate selection of parameters. (Do not attempt to enter braces or to stack parameters.) Braces surrounding stacked items represent alternatives, one of which you must select.<br><br>The example indicates that you must enter FIRST, LAST, or a value for *begin*. | $\left\{\begin{array}{l}\textbf{FIRST}\\ \textbf{\textit{begin}}\\ \textbf{LAST}\end{array}\right\}$ |
| $\underline{\text{Underlining}}$ | Indicates the default value supplied when you omit a parameter.<br><br>The example indicates that if you do not specify ON, OFF, or a row and column destination, the system defaults to ON. | **SCROLL** $\left[\begin{array}{l}\underline{\textbf{ON}}\\ \textbf{OFF}\\ [\textbf{\textit{row}}]\ [,\textbf{\textit{col}}]\end{array}\right]$ |
| | Underlining also indicates an allowable abbreviation or the shortest truncation allowed.<br><br>The example indicates that you can enter either PRO or PROTECTED. | $\underline{\texttt{PRO}}\texttt{TECTED}$ |
| Ellipsis points... | Indicate that the preceding item can be repeated.<br><br>The example indicates that you can enter (A), (A,B), (A,B,C), or some other argument in the same pattern. | `(`*`argument`*`,...)` |
| SMALL CAPS | Represent a keystroke. Multiple keystrokes are hyphenated. | ALT-TAB |

| Convention | Description | Example |
|---|---|---|
| UPPERCASE | Indicates MANTIS reserved words. You must enter them exactly as they appear.<br><br>The example indicates that you must enter CONVERSE exactly as it appears. | `CONVERSE` *`name`* |
| lowercase | Indicates generic names of parameters for which you supply specific values as needed. | `COMPOSE[`*`program-name`*`]` |
| *Italics* | Indicate variables you replace with a value, a column name, a file name, and so on.<br><br>The example indicates that you can supply a name for the program. | `COMPOSE [`*`program-name`*`]` |
| Punctuation marks | Indicate required syntax that you must code exactly as presented.<br><br>`(   )` parentheses<br>`.` period<br>`,` comma<br>`:` colon<br>`'  '` single quotation marks | $\left[\textbf{LET}\right]v\begin{bmatrix}(i)\\(i,j)\end{bmatrix}\left[\textbf{ROUNDED}(n)\right]=e1\ \left[,e2,e3\ldots\right]$ |

## MANTIS documentation series

MANTIS is fourth-generation programming language used for application development. MANTIS is part of AD/Advantage®, which offers additional tools for application development. The following list shows the manuals offered with MANTIS for Windows, organized by task. You may not have all the manuals that are listed here.

**Getting started**

- *MANTIS for Windows Administration Guide*, P19-2304*

**General use**

- *MANTIS for Windows Language Reference Manual*, P19-2302
- *MANTIS for Windows Facilities Reference Manual*, P19-2301
- *MANTIS for Windows Quick Reference*, P19-2303

**SQL support**

- *MANTIS for Windows SQL Support for SUPRA Programming Guide*, P19-2307
- *MANTIS for Windows SQL Support for SUPRA Administration Guide*, P19-2308*

**Master user tasks**

- *MANTIS for Windows Administration Guide*, P19-2304*
- *MANTIS for Windows SQL Support for SUPRA Administration Guide*, P19-2308*

NOTE

Manuals marked with an asterisk (*) are listed twice because you use them for different tasks.

NOTE

MANTIS educational material is available from your regional Cincom education department.

# 1 Introduction

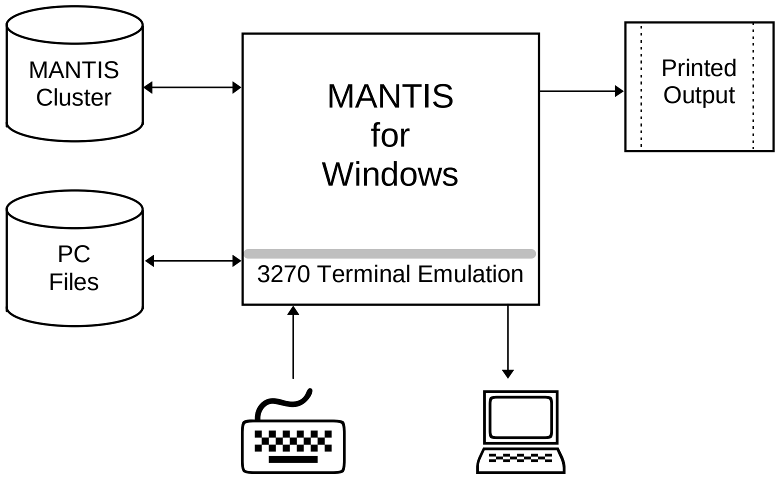
## MANTIS for Windows

MANTIS for Windows is an application development workbench for users of MANTIS on the mainframe, offering the functionality of MANTIS for Windows. MANTIS is composed of design facilities (for creating MANTIS entities such as screens and files) and the MANTIS programming language. This manual discusses the MANTIS programming language used for writing applications. It also explains the conventions of the MANTIS language and explains how MANTIS handles text and numeric data.

MANTIS entities (such as programs, screens, and files) are maintained by MANTIS in its own library. MANTIS entities can be ported to and from the IBM mainframe and copied between two personal computers. They can also be copied to diskettes or printed. MANTIS can also be used to do the following:

- Design and create formatted screens.
- Design and create permanent files for storing and manipulating data.
- Create and test programs interactively. MANTIS provides a compatibility mode for creating programs that are compatible with MANTIS for the IBM mainframe. This mode prevents execution of statements and features not currently supported by MANTIS for the IBM mainframe.

The following figure outlines how MANTIS for Windows works:



## MANTIS for Windows considerations

Considerations for MANTIS for Windows include the following:

♦ This product emulates an IBM 3270-type device to the extent possible with the personal computer hardware. It supports both color displays and monochrome displays (text-mode only).

♦ MANTIS for Windows uses logical keys that correspond to 3270 keys (since PC keyboards do not have all the keys that 3270 keyboards have). Logical keys are special key assignments used to enter data and perform special functions. Personal computer keys are mapped to logical keys, and this mapping can be customized (refer to the *MANTIS for Windows Administration Guide*, P19-2304).

♦ Pressing CTRL-BREAK interrupts the MANTIS program that is currently executing. MANTIS terminates the program and issues a fault message.

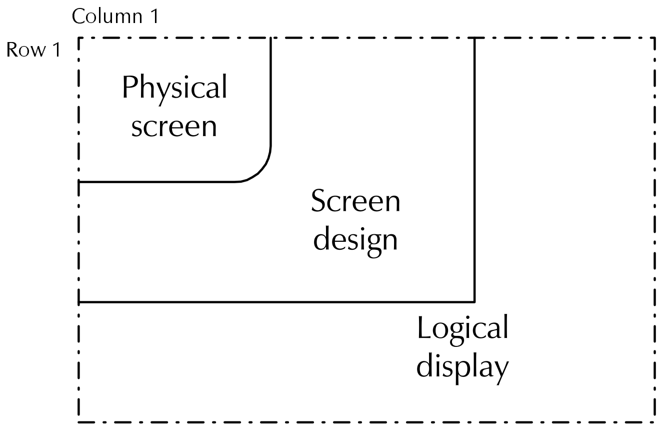
## Terminology used in this document

The following table lists the terms and abbreviations specific to MANTIS for Windows:

| Term | Meaning |
|---|---|
| Compatibility mode | Creates programs and specifies attributes compatible with MANTIS for the IBM mainframe. Prevents execution of statements or features not currently supported by MANTIS for the IBM mainframe. Features allowed in compatibility mode may not be available in MANTIS for the IBM mainframe if you do not have the current release. |
| DOS | MS-DOS |
| Logical keys | Special key assignments that correspond to 3270 keys, enter data, and perform special functions. |

## Data display modes

MANTIS performs terminal I/O through the Logical Terminal Interface (LTI), which supports a logical display area of 32,767 rows by 32,767 columns. Screens that are larger than the physical screen can also be created. The physical screen acts as a window through which the contents of the logical display can be viewed, as illustrated by the following figure:



The physical screen can be moved around the logical display using the MANTIS windowing keys (see the table under "MANTIS editing and windowing keys" on page 32).

MANTIS uses two methods to display data on the logical display. The description of these methods, full-screen mode and scroll mode, provides a brief explanation of the CONVERSE, SHOW, OBTAIN, and HEAD statements. These statements are described in detail in "Statements and built-in functions," beginning on page 65.

NOTE

If you are a new MANTIS user, you may want to skip this section and become more familiar with MANTIS screen design and the CONVERSE, SHOW, OBTAIN and HEAD statements first.

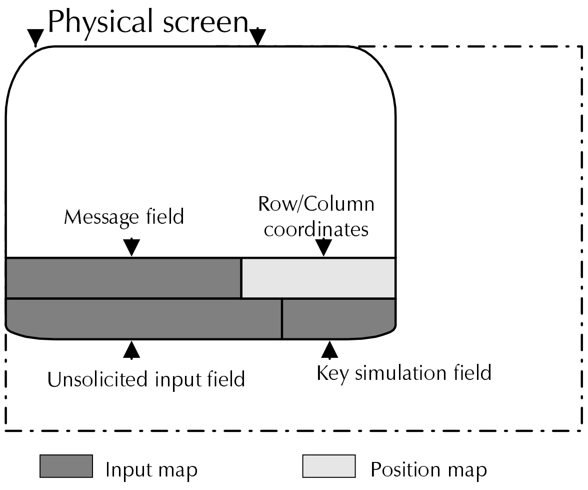
## Full-screen mode

Full-screen mode describes the way MANTIS displays a screen (also called a map). First, use the MANTIS Screen Design Facility to create your screen, and define its layout, content, and attributes. Then, use the CONVERSE statement (in your MANTIS program) to identify the screen you want to display, and its position on the logical display, and ultimately on your physical screen. When MANTIS converses (displays) your screen (also referred to as a map), it waits for you to inspect it or enter data. When you press ENTER, MANTIS processes any data you have entered according to subsequent program statements.

In full-screen mode, MANTIS displays the following floating maps:

- **Input map**. Contains fields for messages and user-entered data.
- **Position map**. Contains row and column coordinates for the window position.

MANTIS uses floating maps to display the fields in the following figure:



In addition to user-defined maps, MANTIS uses several built-in maps to provide additional terminal controls.  Some of these maps are floating maps that MANTIS displays.  Floating maps have a constant position relative to the physical screen in the logical display.  When you move the physical screen, the floating map moves in the same direction, retaining its position on the physical screen.

The following table lists and explains the fields that MANTIS displays in full-screen mode:

| Field | Length | Function |
|---|---|---|
| Message field (input map) | Terminal width, less 12 columns | Displays MANTIS messages. |
| Row/Column coordinates (position map) | 5 bytes for each | Displays current row and column coordinates of the upper left corner of the window in the logical display.  Press WINDOWMAP to add or remove this field.  When the position of the window is displayed, enter new values in the row and column fields.  When you press ENTER, MANTIS interprets the values as follows:<br>♦ **Unsigned value.** Row or column number in the logical display.<br>♦ **Signed value.** Row or column number relative to the current value.<br>♦ **I precedes value.** Row or column increment for window movement keys. |
| Unsolicited Input field (input map) | Terminal width, less 8 column | Provides field for entering unsolicited input.  Add or remove this field by pressing INPUTMAP in full-screen mode. |
| Key Simulation field (input map) | 6 bytes | Provides field for entering function keys.  This value is obtained using the screen-name function in your MANTIS program.  Add or remove this field by pressing INPUTMAP. |

The last two lines may not appear on your conversed map.  You can specify attributes in screen design that tell MANTIS to display your map over the last two lines, or to protect the fields in the bottom line.

The CONVERSE statement can be used to display more than one map on the logical display. A collection of maps on the logical display is called a map set. The order in which maps are added to the map set (conversed) determines their order in the display as well as whether they are active (fields are updatable) or passive (fields are not updatable) in the display. "MANTIS Logical Terminal Interface" on page 251 discusses the CONVERSE statement and map sets in more detail.

## Scroll mode

MANTIS automatically creates a scroll map and uses scroll mode to display data when a MANTIS program uses the following statements:

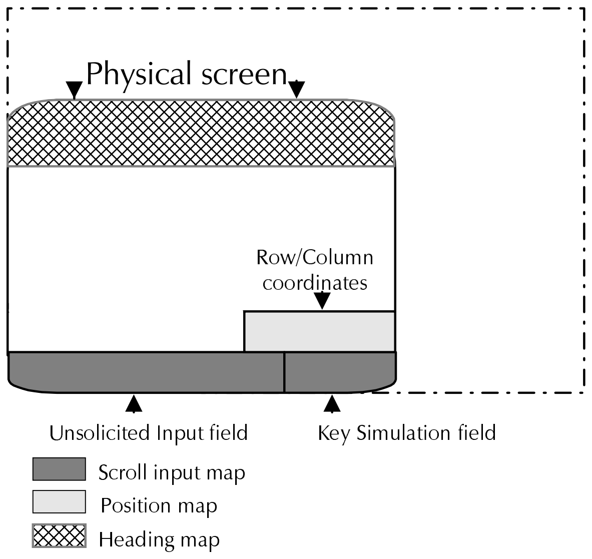
- **SHOW**. Specifies the data you want to display on your physical screen.
- **OBTAIN**. Typically follows SHOW in a MANTIS program, and gets the response to the SHOW data.
- **HEAD**. Provides a centered heading for the display. The HEAD statement always reserves two lines on your display:
  - One line for the heading.
  - The other line for a blank line to separate the heading from the text on the screen.

MANTIS displays each new line of data at the bottom of the scroll map, and scrolls up previous lines one line at a time. Scroll mode is also used by the Update Program option of the Program Design Facility. When the scroll mode map fills with lines, MANTIS displays MORE in the Key Simulation field and waits for you to press ENTER. This may occur, for example, if multiple SHOW statements are issued without an intervening OBTAIN or WAIT statement, or when using Program Design commands such as LIST, USAGE, and so on.

The floating maps MANTIS displays in scroll mode are the following:

- **Heading.** Displays a screen heading.
- **Position.** Contains row and column coordinates for the physical screen position.
- **Scroll input.** For user-entered data.

These fields are shown in the following figure:



The following table lists and explains the fields that MANTIS displays in scroll mode:

| Field | Length | Function |
|---|---|---|
| Unsolicited Input field (input map) | Terminal width, less 8 columns | Provides field for entering input (e.g., response to a SHOW and OBTAIN). |
| Row/Column coordinates (position map) | 5 bytes for each | Displays current row and column coordinates of the upper left corner of the window in the logical display. Press WINDOWMAP to add or remove this field. When the position of the window is displayed, enter new values in the row and column fields. When you press ENTER, MANTIS interprets the values as follows: ♦ **Unsigned value.** Row or column number in the logical display. ♦ **Signed value.** Row or column number relative to the current value. ♦ **I precedes value.** Row or column increment for window movement keys. |
| Key Simulation field (input map) | 6 bytes | Provides field for entering function key values. This value is obtained using the KEY function in your MANTIS program. Add or remove this field by pressing INPUTMAP. |

NOTE

Although the physical screen is a window on the scroll map, some data that is displayed may not be part of the scroll map. Output may be directed to your screen by the operating system rather than by MANTIS. In this case, pressing SELECT will not copy what is on your screen. To see what is copied from the scroll map, press REFRESH before pressing SELECT (see the table in “MANTIS editing and windowing keys” on page 32).

## KILL and CTRL-BREAK

KILL is a special value that can be entered in the Key Simulation field of a formatted or unformatted screen. KILL terminates a program listing, a program waiting for keyboard input, or a program in a loop. Whenever a program executes a WAIT, OBTAIN, or CONVERSE statement, or has just issued the message “POTENTIAL PROGRAM LOOP ENCOUNTERED”, enter KILL to stop program execution.

When MORE is displayed in the Key Simulation field, you can overtype MORE with KILL if you want to terminate your program or the Program Design command.

You can reexecute your program by issuing the RUN statement if you are in programming mode.

You can use the KILL macro key (see the table in the “MANTIS macro keys” section on page 36) instead of typing KILL in the Key Simulation field.

CTRL-BREAK can also be used to terminate a program. The differences between KILL and CTRL-BREAK are:

| **Area** | **KILL** | **CTRL-BREAK** |
|---|---|---|
| Behavior | Causes the program to behave as if a STOP statement was encountered at that point. | Causes the currently executing statement to fault. |
| Use | Can be used only when a program is waiting for input. | May be used at any time. |
| Trapping | Cannot be trapped. | May be trapped via a fault handling routine (see “SET” on page 202). |

## Windowing

Your physical terminal display acts as a window for the logical terminal display. You can move this window around the logical display using the function keys listed in the table in "Scroll mode," beginning on page 25. When this window is moved, your screen is updated to show the contents of the logical display at the new window position. You may notice that some parts of your screen remain unchanged. This can happen for one of two reasons:

- To maintain compatibility with MANTIS for the IBM mainframe, MANTIS for Windows uses a field separator to simulate the invisible attribute byte that precedes every field on an IBM 3270 terminal. When a map is designed using the Screen Design Facility, one position in front of every field is reserved by default for the field separator. As a result, column one in the display is always blank for fields designed with field separators. For maps designed without field separators, column one is not reserved.
- MANTIS can modify the screen layout by adding floating maps.

## Keyboard operation

Logical key names are used to describe keyboard operations. Logical keys correspond to 3270 keys or special MANTIS terminal control functions. A standard personal computer has the following types of keys:

- A main keypad of alphabetic (a–z), numeric, and punctuation data-entry keys.
- Special keys such as function keys, TAB, ENTER, directional cursor keys, HOME, END, and PGUP.
- ALT and CTRL, that are used like the SHIFT key, and held down while another key is pressed.

Use the keyboard to do the following:

- **Enter data.** Data can only be entered when the cursor is in an unprotected field on the screen. When you press a data-entry key, the corresponding character is entered into the field at the cursor position, and the cursor moves one position to the right. If you are in overtype mode (the default mode), each character that is entered overwrites the character at the cursor position. In insert mode (turned on/off by the INSERT key), each character that is entered is inserted at the cursor position, and the following characters in the field move one position to the right.
- **Perform special functions.** When you press one of the keys listed in the table in “MANTIS editing and windowing keys” on page 32, MANTIS performs a function that has an immediate effect on your screen (such as moving the window or initiating insert mode) without returning control to the application.

You can also use logical keys, such as those listed in the table in the "MANTIS action keys" section on page 35 or the table in the "MANTIS macro keys" section on page 36, to return control to the MANTIS program that is executing. The program can obtain the name of the function that you selected. The response of the logical key is determined by the logic of the MANTIS program. You can also enter any value in the Key Simulation field before pressing ENTER, and this value will be returned to the MANTIS program as the logical key name. Entering KILL in the Key Simulation field, however, terminates the MANTIS program.

The following tables show the default key assignments you can use when running your application. Logical key names (such as CANCEL and INSERT) are used throughout this manual. The personal computer key values can be changed in keyboard configuration (refer to the *MANTIS for Windows Administration Guide*, P19-2304).

## MANTIS editing and windowing keys

The following table shows the keys that you use to move the cursor and the window and modify the screen, without sending any data from the input screen to the MANTIS application program:

| Logical key | PC key | Enables you to |
|---|---|---|
| UP | ↑ | Move the cursor up one row. |
| DOWN | ↓ | Move the cursor down one row. |
| LEFT | ← | Move the cursor left one column. |
| RIGHT | → | Move the cursor right one column. |
| TAB | →\| (TAB) | Move the cursor to the start of the next unprotected field. |
| BACKTAB | SHIFT-→\| (SHIFT-TAB) | Move the cursor to the start of the current unprotected field (or to the start of the previous unprotected field if already at the start of the current field). |
| DELETE | DEL | Delete the character at the cursor. |
| BACKSPACE | ← (BACKSPACE) | Delete the character to the left of the cursor and move the cursor one space to the left. |
| ERASEEOF | F6 or CTRL-END | Delete the character(s) from the cursor to the end of the field. |
| INSERT | INS | Turn on/off insert mode. |
| HOME | HOME | Move the cursor to the first unprotected field on the screen. |

| **Logical key** | **PC key** | **Enables you to. . .** |
|---|---|---|
| NEWLINE | CTRL-ENTER | Move the cursor to the first unprotected field on the next line. |
| REFRESH[1] | CTRL-R | Redisplay the current screen. |
| SELECT[1] [2] | CTRL-S | Copy a line from the scroll map to the Unsolicited Input field for you to modify. |
| SELUP[1] | CTRL-U | Copy the previous input line from the scroll map to the Unsolicited Input field. |
| SELDOWN[1] | CTRL-D | Copy the next input line from the scroll map to the Unsolicited Input field. |
| TABWRD | CTRL-→ | Move the cursor to the next word. |
| TABBOW | CTRL-← | Move the cursor to the beginning of the word (\or to the previous word if the cursor is already at the beginning of a word). |
| TABEOF | END | Move the cursor past last nonblank character in current field. |
| LINEDRAW[3] | CTRL-B | Draw lines in Screen Design by using the cursor keys. |
| LINECLEAR[3] | CTRL-N | Erase line-drawing characters in Screen Design by using the cursor keys. |
| STOP | CTRL-\ | Terminate MANTIS.  You must press this key twice, consecutively. |
| WINUP | PGUP | Move the window up by the row increment value. |

1 This key is used in the MANTIS Line Editor.  Refer to the *MANTIS for Windows Facilities Reference Manual*, P19-2301, for more information.

2 See “PERFORM” on page 178 for more information on using SELECT.

3 This key remains active until you press an action key (see “MANTIS action keys” on page 35).

| **Logical key** | **PC key** | **Enables you to. . .** |
|---|---|---|
| WINDOWN[1] | PGDN | Move the window down by the row increment value. |
| WINLEFT[1] | CTRL-PGUP | Move the window left by the column increment value. |
| WINRIGHT[1] | CTRL-PGDN | Move the window right by the column increment value. |
| WINTOPL | CTRL-HOME | Move the window to the top left of the logical display. |
| WINTOPR | No default | Move the window to the top right of the logical display. |
| WINBOTL | No default | Move the window to the bottom left of the logical display. |
| WINBOTR | No default | Move the window to the lower-right of the logical display. |
| SCROLLALL | CTRL-A | Display the entire scroll output map by scrolling from top to bottom. |
| INPUTMAP | CTRL-I | Add or remove the Unsolicited Input field, Key Simulation field and Message field. |
| WINDOWMAP | CTRL-W | Add or remove the row/column coordinates at the bottom of the screen. |
| STATUSLINE | CTRL-P | Add or remove the status line on the last line of the screen.  The status line shows the current time, insert mode, and cursor position. |
| VALIDINFO | CTRL-V | Display extended edit attributes for a particular field. Position cursor over the field and press VALIDINFO anytime during data entry to the screen. |

1  See the tables in sections “Full-screen mode” on page 22 and “Scroll mode” on page 25 to change the increment value.

## MANTIS action keys

The following table shows the MANTIS action keys you use to send data from the input screen to the MANTIS application program. The function of each key is determined by the application. It returns the logical key name to the map variable in the application program. Data fields are modified to match the screen fields.

| Logical key | PC key | Logical key | PC key |
|---|---|---|---|
| CANCEL | ESC | PF11 | ALT-- |
| CLEAR* | F2 | PF12 | ALT-+ |
| ENTER | ENTER | PF13 | ALT-Q |
| PA1 | ALT-J | PF14 | ALT-W |
| PA2 | ALT-K | PF15 | ALT-E |
| PA3 | ALT-L | PF16 | ALT-R |
| PF1 | ALT-1 | PF17 | ALT-T |
| PF2 | ALT-2 | PF18 | ALT-Y |
| PF3 | ALT-3 | PF19 | ALT-U |
| PF4 | ALT-4 | PF20 | ALT-I |
| PF5 | ALT-5 | PF21 | ALT-O |
| PF6 | ALT-6 | PF22 | ALT-P |
| PF7 | ALT-7 | PF23 | ALT-A |
| PF8 | ALT-8 | PF24 | ALT-S |
| PF9 | ALT-9 | | |
| PF10 | ALT-0 | | |

* This key erases the entire screen.

## MANTIS macro keys

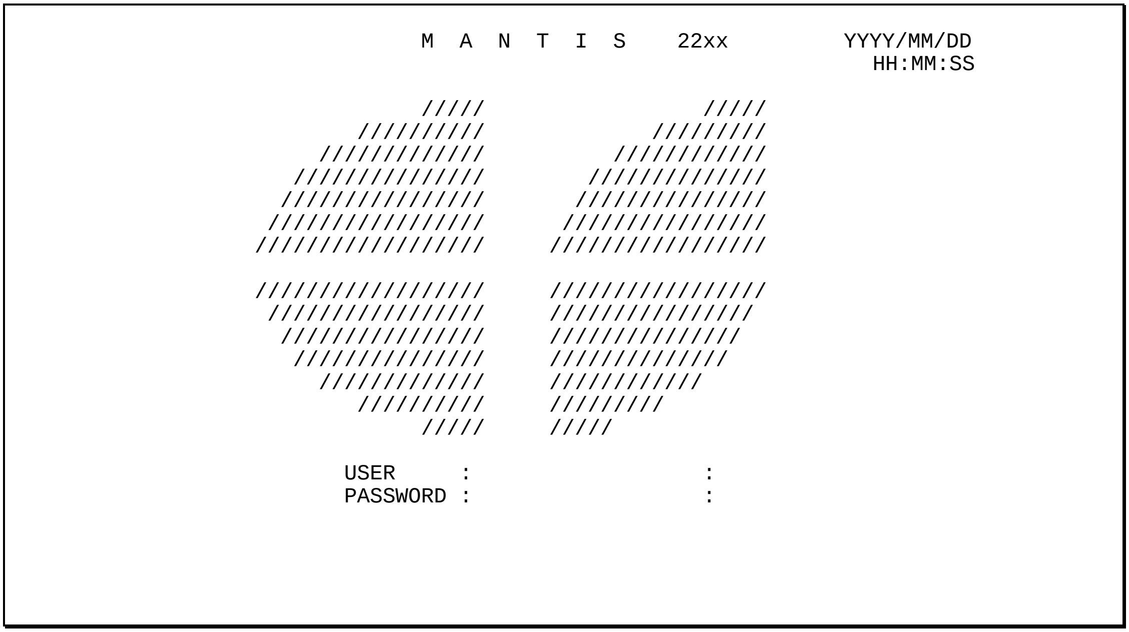
The following table shows the keys that invoke an internal function that is equivalent to typing the logical key name in the Key Simulation field and pressing ENTER:

| Logical key | PC key | Enables you to. . . |
|---|---|---|
| EDIT* | CTRL-E | Invoke the user-specified editor. |
| HELP* | CTRL-H | Display help prompter. |
| KILL | CTRL-K | End program loop. |
| QUIT* | CTRL-Q | Exit from programming mode and return to the Program Design Facility menu. |

* The function of this key is application-dependent. The description here is the intended purpose of the key. Consult each facility for details.

## Signing on to MANTIS

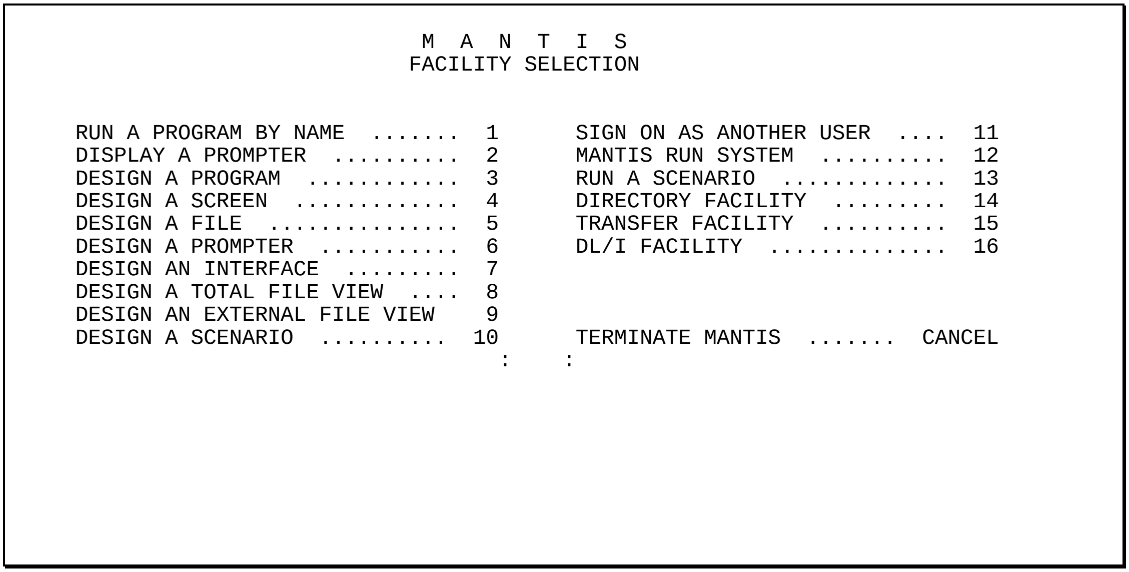
When you access MANTIS for Windows, the following sign-on screen displays:



```
             M  A  N  T  I  S    22xx         YYYY/MM/DD
                                                HH:MM:SS

             /////                 /////
        //////////             /////////
     /////////////          ////////////
   ///////////////        //////////////
  ////////////////       ///////////////
 /////////////////      ////////////////
//////////////////     /////////////////

//////////////////     /////////////////
 /////////////////     ////////////////
  ////////////////     ///////////////
   ///////////////     //////////////
     /////////////     ////////////
        //////////     /////////
             /////     /////

       USER     :                  :
       PASSWORD :                  :
```

To sign on to MANTIS, enter your user name and password and press ENTER. You must enter your password exactly as it appears in your User Profile (uppercase or lowercase). The Facility Selection screen displays and you are signed on to MANTIS.

The following screen illustrates the standard facilities provided with MANTIS. To request a facility from the menu, either enter the number corresponding to the facility you want in the Action field (: :) and press ENTER or press the corresponding PF key.



```
                                M  A  N  T  I  S
                               FACILITY SELECTION


     RUN A PROGRAM BY NAME  .......  1      SIGN ON AS ANOTHER USER  ....  11
     DISPLAY A PROMPTER  ..........  2      MANTIS RUN SYSTEM  ..........  12
     DESIGN A PROGRAM  ............  3      RUN A SCENARIO  .............  13
     DESIGN A SCREEN  .............  4      DIRECTORY FACILITY  .........  14
     DESIGN A FILE  ...............  5      TRANSFER FACILITY  ..........  15
     DESIGN A PROMPTER  ...........  6      DL/I FACILITY  ..............  16
     DESIGN AN INTERFACE  .........  7
     DESIGN A TOTAL FILE VIEW  ....  8
     DESIGN AN EXTERNAL FILE VIEW    9
     DESIGN A SCENARIO  ..........  10      TERMINATE MANTIS  .......  CANCEL
                                      :    :
```

When you exit from one of the facilities, you always return to this menu where you can select another facility or exit from MANTIS.

To exit MANTIS from the Facility Selection menu, press CANCEL.

# 2

# MANTIS language conventions

## MANTIS programming guidelines

This chapter explains the conventions the MANTIS language uses. This chapter is not intended to teach you *how* to program, but rather outlines the guidelines that MANTIS uses so you can apply them to your current data processing background.

Throughout this chapter, short program examples illustrate specific aspects of MANTIS programming. You do not have to understand each statement at this time. "Statements and built-in functions" on page 65 provides more details on statements and functions. To create your MANTIS programs, you use the MANTIS Program Design Facility (refer to the *MANTIS for Windows Facilities Reference Manual*, P19-2301), which explains how to access the facility and use the MANTIS editing commands. This manual explains the MANTIS programming language.

## Character set

MANTIS programs are written in the MANTIS language.  To use MANTIS, you must know some fundamental rules about how MANTIS works.  The character set consists of:

- Alphabetic characters A–Z (uppercase), a–z (lowercase).
- Space character (blank).
- Numbers (0–9).
- Special characters (defined in the following table).

| Character | Meaning |
|---|---|
| “ | Double quotes enclose a text literal. |
| ‘ | A single quote (apostrophe) marks a continuation line. |
| ( ) | Parentheses enclose subscripts, subexpressions, arguments, and parameters. |
| : | A colon separates two program statements on the same line.  A colon also abbreviates the SHOW command, as refer to the *MANTIS for Windows Facilities Reference Manual*, P19-2301. |
| ; | A semicolon inserts a space in a line displayed by the SHOW statement. |
| , | A comma separates subscripts, arguments, and parameters.  It also inserts spaces up to the next zone in a line displayed by the SHOW statement. |
| . | A period designates the decimal point in a number. |
| _ | An underline may be used in a symbolic name. |
| ¦ | A broken vertical bar precedes a comment in a program line. |
| + | A plus sign adds one number to another or joins one text value to another. |
| - | A minus sign subtracts one number from another or reduces one text value by another. |

| Character | Meaning |
|---|---|
| * | An asterisk multiplies one number by another. |
| ** | Two asterisks raises one number to the power of another. |
| / | A slash divides one number by another. |
| = | An equals sign tests whether one value is equal to another, or assigns a value to a variable. |
| < | A less-than sign tests whether one value is less than another. |
| > | A greater-than sign tests whether one value is greater than another. |
| < > | A less-than sign followed by a greater-than sign tests whether one value is unequal to another. |
| >= | A greater-than sign followed by an equals sign tests whether one value is greater than or equal to another. |
| <= | A less-than sign followed by an equals sign tests whether one value is less than or equal to another. |
| & | An ampersand indicates an indirect reference to a symbolic name. (See "Indirect reference (&)" on page 63.) |
| @ | An at sign directs MANTIS to obtain keyboard input from a DOS file (programming mode only). |
| $ | A dollar sign is short for the PERFORM command. (The dollar sign can be used as a command only, not in a programming statement.) |

## Reserved words

A reserved word is a word that has a special meaning to MANTIS. MANTIS uses reserved words to identify commands and statements and their associated options.  These reserved words also identify built-in functions and constants.  They must be entered exactly as they appear and cannot be used for any other purpose.

The following table lists reserved words:

| | | | | |
|---|---|---|---|---|
| ABS | ACCESS | AFTER | ALL | ALTER |
| AND | ASI | AT | ATN | ATTRIBUTE |
| BEFORE | BIG | BIND | BREAK | BY |
| CALL | CHAIN | CHANGE | CHR | CLEAR |
| COMMIT | CONVERSE | COPY | COS | CURSOR |
| DATAFREE | DATE | DBPAGE | DELETE | DEQUEUE |
| DISPLAY | DO | DOLEVEL | DOWN | |
| E | EDIT | ELSE | END | ENQUEUE |
| ENTRY | EQUAL | ERASE | EXEC_SQL | EXECUTE |
| EXIT | EXP | | | |
| FALSE | FILE | FIRST | FOR | FORMAT |
| FSI | | | | |
| GET | GO | | | |
| HEAD | HELP | | | |
| IF | INSERT | INT | INTERFACE | INTERNAL |
| KANJI | KEY | | | |
| LANGUAGE | LAST | LET | LEVEL | LIST |
| LOAD | LOG | | | |

| | | | | |
|---|---|---|---|---|
| MARK | MEMORY | MIXED | MODIFIED | |
| NEW | NEXT | NOT | NULL | NUMERIC |
| OBTAIN | OFF | ON | OR | ORD |
| OUTPUT | | | | |
| PAD | PASSWORD | PERFORM | PERM | PI |
| POINT | POSITION | PREFIX | PRINTER | PRIOR |
| PROGRFREE | PROGRAM | PROMPT | PURGE | |
| QUIT | | | | |
| RELEASE | REPLACE | RESERVED | RESET | RETURN |
| RND | ROUNDED | ROUNDING | RUN | |
| SAME | SAVE | SCREEN | SCROLL | SEED |
| SELECT | SEQUENCE | SET | SQN | SHOW |
| SIN | SIZE | SLICE | SLOT | SMALL |
| SPECTRA | SQLBIND | SQLCA | SQLDA | SQR |
| STOP | SUBMIT | $SYMBOL | | |
| TAN | TERMINAL | TERMSIZE | TEXT | TIME |
| TO | TOTAL | TRAP | TRUE | TXT |
| ULTRA | UNPAD | UNTIL | UP | UPDATE |
| UPPERCASE | USAGE | USER | USERWORDS | |
| VALUE | VIA | VIEW | VSI | |
| WAIT | WHEN | WHILE | WINDOW | |
| ZERO | | | | |

## Symbolic names

A symbolic name is a string of characters that represents a user-defined object (such as a screen or field) in a MANTIS program. MANTIS uses symbolic names to represent variables processed by a MANTIS program. Symbolic names can stand for either numeric or text data. MANTIS allows a maximum of 65,536 symbolic names for a single program, including names defined indirectly by SCREEN, FILE, ACCESS, and INTERFACE statements. The Master User sets the maximum value in the configuration file. (Note, however, that memory limitations will prevent this release of MANTIS from ever reaching this maximum.)

The following considerations apply to a symbolic name:

- Must begin with an alphabetic character.
- Can contain alphabetic characters, numeric characters, and the underline (_). No other special characters are allowed in a symbolic name. Lowercase characters can be entered; MANTIS will convert them to uppercase (e.g., the following variables are equivalent: customer_name, Customer_Name, CUSTOMER_NAME).
- Must not be a reserved word, as listed in the table in the "Reserved words" section, beginning on page 42. A symbolic name can contain a reserved word (e.g., EDITOR) but cannot be a reserved word in its entirety (e.g., EDIT).
- Has a maximum length of 80 characters.

## Variables and arrays

MANTIS supports the following variable types:

- **TEXT**. Text (ASCII) character strings.
- **SMALL**. Floating-point numbers with up to 6 significant digits.
- **BIG**. Floating-point numbers with up to 15 significant digits.

Variables are created dynamically during program execution and may be defined in one of the following ways:

- Explicitly through a TEXT, SMALL, or BIG statement.
- Implicitly through use of a symbolic name either within an expression or as the receiving variable in a LET statement. When defining variables implicitly, MANTIS defines the variable as BIG with a value of zero.
- Indirectly through a FILE, SCREEN, or ACCESS statement.

Arrays are ordered sets of values. Each value is called an array element. Arrays can have from 1–255 dimensions (as specified by your Master User in the configuration file), with each dimension in the range 1–16000. Memory required for an array must not exceed 64K.

You can define arrays with a BIG (15 significant digits), SMALL (6 significant digits), or TEXT statement or with a SCREEN, FILE, or ACCESS statement if these definitions contain implied arrays.

To access one element of an array, specify its position within the array by subscripting the array variable. A subscript can be either a number or an arithmetic expression. For example, if you have an array with 17 elements:

```
10 BIG STAR(17)
```

You can access the first element by entering STAR(1); you can access the eleventh element in any of the following ways:

```
STAR(11)   or   K=11        or   STAR(6+5)   or   STAR(1)=9
                STAR(K)                           K=2
                                                  STAR(STAR(1)+K)
```

The value you use to subscript an array must be between one and the maximum dimension given for that array.  If the value is outside this range, MANTIS terminates the program with an error message.  If, for example, you specify:

```
SMALL NEUTRON(7,3)
NEUTRON(6,4) = 2
```

MANTIS displays an error message because the second dimension (4) exceeds the maximum (3) defined in the previous statement.

When specifying TEXT arrays, the last dimension is the maximum length of each element.  The following example defines a three-dimensional array where each element has a maximum length of 10 characters:

```
TEXT XYZ(3,2,5,10)
```

# Language fundamentals

This section describes the following components of the MANTIS language:

- Statements
- Commands
- Program lines
- Comments

## Statements

A statement is the smallest unit of execution within MANTIS and consists of a reserved word and, if necessary, one or more operands. A statement can be part of a program line (saved with the program) or can be issued for immediate execution within programming mode (not saved with the program). All statements are described in "Statements and built-in functions," beginning on page 65.

You can use spaces freely when entering a statement. MANTIS ignores (and discards) all spaces except when they appear in text literals or perform a delimiting function. For example, the following statements have the same meaning to MANTIS:

```
   SHOW     ALPHA ,    BETA,       -1500+ 7
 SHOW   ALPHA,BETA     , -1500   + 7
```

## Commands

A command is a special form of a statement that can be issued only for immediate execution within programming mode. MANTIS commands are discussed in *MANTIS for Windows Facilities Reference Manual*, P19-2301.

## Program lines

A program line consists of a line number (1–64000) followed by one or more statements.  Multiple statements on one program line must be separated by a colon (:).  For example:

```
20 CLEAR:HEAD "This heading appears at the top of the screen"
```

You cannot include a logic statement on a program line with another statement.  A logic statement must appear on a line by itself.  The following table lists the logic statements:

| | |
|---|---|
| BREAK | FOR |
| CHAIN | IF |
| DO | NEXT |
| ELSE | RETURN |
| END | UNTIL |
| ENTRY | WHEN |
| EXEC_SQL | WHILE |
| EXIT | |

Statements can contain as many characters as will fit on the line.  If you need more characters, continue the line by entering an apostrophe (‘) immediately after the next line number.  For example:

```
20 "THE RELATIVE HUMIDITY IS"; HMD; "AND THE TEMPERATURE IS";
16 'TEMP
```

Do not separate a symbolic name (such as TEMP), reserved word, or a number (such as 12003).  Both of the following continuation attempts are incorrect.  In the first example, instead of showing the value of TEMP, two values would be shown (TE and MP).  If there are not any fields TE and MP, then fields will be defined by default with these names as BIG variables.  In the second example, 12003 is treated as two separate numbers, 12 and 003.

```
15 SHOW "RELATIVE HUMIDITY IS": HMD; "AND TEMPERATURE IS"; TE
16 'MP
17 SHOW "RELATIVE HUMIDITY IS"; HMD; "AND POLLEN COUNT IS";12
18 '003
```

If you have to continue a long literal (text enclosed in quotes), close the literal on the first line (closing quotation mark (”)) and reopen it (opening quotation mark (“)) on the continuation line:

```
30 SHOW "THIS IS A WEATHER REPORT FOR ALL REGIONS SOUTH AND "
35 '"SOUTHWEST OF THE VALLEY"
```

## Comments

A comment is a statement that consists of a broken vertical bar ( ¦ ) and valid text characters. MANTIS assumes that all characters up to the end of the line (including any colon (:)) are part of the comment, so a comment must be the last (or only) statement on a line. A comment cannot be continued onto another program line. For example:

```
10  ¦ This is a valid comment
20 . SHOW "WHAT IS THE CAPITAL AMOUNT?": ¦ Here is another valid
  comment
30 . ¦ Here is an example of an
40 .'Invalid comment
```

## Text considerations

This section discusses how MANTIS stores and manipulates text data. MANTIS stores text data in TEXT variables. Each TEXT variable can hold a maximum of 32750 characters. The Master User specifies the maximum value in the configuration file. Any character in the ASCII character set can be stored in a TEXT variable.

### Storing text data

MANTIS stores text data as either literals or variables. A text literal is any string of characters enclosed in quotes (e.g., “January 1st, 1998”). Quotes within a text literal must be duplicated to distinguish them from the enclosing quotes. For example:

```
SHOW "THE ANSWER IS ""X""."
```

produces

```
THE ANSWER IS "X".
```

You define a TEXT variable with a TEXT statement or with a SCREEN, FILE, or ACCESS statement that indirectly defines TEXT variables. The default for a variable is BIG, so you must define TEXT variables before using them. If you define two text variables as:

```
TEXT ALPHA(20), BETA(14,20)
```

MANTIS allocates 20 characters of storage for ALPHA. Since BETA is a one-dimensional array of text variables, MANTIS allocates memory for 14 elements where each element can have a maximum of 20 characters.

A text variable or array element cannot exceed 32750 characters (or the maximum defined by the Master User in the configuration file). It has these length characteristics:

- A maximum length as defined in the TEXT statement.
- A current length as maintained by MANTIS. The current length indicates how many characters the text variable currently contains.

For example, if you enter:

```
TEXT ALICE(13,20)
```

MANTIS creates a text array with 13 elements, each with a defined maximum length of 20 characters and a current length of zero characters. Then if you execute:

```
ALICE(6) = "TWAS BRILLIG"
```

The sixth element in the array has a current length of 12 while all the other elements still have a current length of zero.

You can use the SIZE function to obtain the current length of a text variable. For example:

```
 SIZE(ALICE(6))
```

returns the value 12.

## Text substrings

The value of a text variable or array element is a string of characters numbered in ascending sequence from left to right. MANTIS allows you to refer to a substring by using subscripts to specify the first and last character positions. If the second subscript is omitted, the substring includes all characters up to the end of the string. Any character positions you specify that fall outside the string will not be included in the substring. For example:

```
TEXT WORDS(20)
  WORDS = "LEND ME YOUR EARS"
```

| | | |
|---|---|---|
| `WORDS(1,17)` | refers to | `LEND ME YOUR EARS` |
| `WORDS(9,12)` | refers to | `YOUR` |
| `WORDS(9)` | refers to | `YOUR EARS` |
| `WORDS(9,19)` | refers to | `YOUR EARS` |
| `WORDS(18)` | refers to | `null string` |

You also can use negative subscripts to refer to a character position relative to the end of the string:

| | | |
|---|---|---|
| `WORDS(-17,-1)` | refers to | `LEND ME YOUR EARS` |
| `WORDS(-12,-11)` | refers to | `ME` |
| `WORDS(-12)` | refers to | `ME YOUR EARS` |
| `WORDS(-19,-11)` | refers to | `LEND ME` |
| `WORDS(-19)` | refers to | `LEND ME YOUR EARS` |

Subscripts do not have to be integers. MANTIS truncates any fraction, as if the INT function had been used.

If you refer to a substring in a text array, the position subscript(s) comes after the dimension subscript(s), which specifies the array element. For example, if ALICE(6) = “TWAS BRILLIG”:

| | | |
|---|---|---|
| `ALICE(6,1,3)` | refers to | `TWA` |
| `ALICE(6,-6,-2)` | refers to | `RILLI` |

You can assign a value to a substring of a text variable or array element by using a LET statement. The substring specification is then interpreted as if the string extended to the maximum length of the text variable or array element. For example:

| | | |
|---|---|---|
| `WORDS(18)="."` | sets WORDS to | `LEND ME YOUR EARS.` |
| `WORDS(9)="TEN DOLLARS"` | sets WORDS to | `LEND ME TEN DOLLARS` |

A subscript can be an arithmetic expression. For example, if you enter:

```
SMALL BLOCK(3,3)
TEXT DANDY(6,14)
BLOCK(1,2) = 1
BLOCK(2,3) = 2
BLOCK(3,1) = 3
BLOCK(3,3) = 5
DANDY(6) = "DRESS WELL"
SHOW DANDY((BLOCK(3,1)*BLOCK(2,3)), BLOCK(1,2),
'  4+BLOCK(3,3))
```

The SHOW statement will go through the following internal computational steps:

```
SHOW DANDY((3*2),1,4+5)
SHOW DANDY(6,1,9)
```

which results in:

```
DRESS WEL
```

## Text expressions

A text expression in MANTIS consists of one or more text operands. The following table lists the various types of text operands with an example of each type:

| Operand | Example |
|---|---|
| Text variable | NOUN,VERB |
| Text array element | ALICE(6) |
| Text substring | WORDS(-8,-3) |
| Text literal | “Dollars”, “50” |
| Built-in text function | TXT(PI), DATE |
| Subexpression in parentheses | (KEY+“PRESSED”) |

The simplest type of text expression is an operand by itself (e.g., NOUN). In expressions with more than one operand, each pair of operands is separated by an operator (e.g., NOUN+VERB). You can join two text operands by using the plus (+) operator. For example, if you enter:

```
TEXT ONE(12), THREE(24), TWO(12)
ONE = "RAIN IS "
TWO = "FALLING"
THREE = ONE+TWO
```

THREE contains:

```
RAIN IS FALLING
```

You can use the minus (-) operator to remove the characters in the second operand from the first operand. In the example above, if you change the assignment of THREE to:

```
THREE = TWO-"LL"
```

THREE will then contain:

```
FAING
```

MANTIS removes only the first occurrence of the characters. You can find the point where the characters were removed by using the text expression as the argument of the built-in text function POINT (see “Relational and logical expressions” on page 60). POINT returns a numeric value. For example, if you execute:

```
A = POINT(TWO-"LL")
```

MANTIS sets A to a value of 3 because “LL” begins in the third position of TWO.

## Numeric considerations

MANTIS numbers can consist of:

- The digits 0–9.
- A preceding plus or minus sign.
- A period to designate the decimal point.
- The letter E to indicate an exponent.

### Storing numeric data

This section discusses how MANTIS stores and manipulates numeric data. MANTIS stores numeric data in a floating-point format, but you will usually see it displayed in standard decimal notation. In some cases, however, MANTIS displays the number in scientific notation with an exponent.

### Scientific notation

The letter E in scientific notation means "times 10 to the power of." The number before the E is a decimal number (with one digit before the decimal point) containing the significant digits of the number. The number after the E (the exponent) is an integer that specifies how many places the decimal point will shift (to the right if positive, to the left if negative) to obtain the actual number. The following table gives examples of scientific notation and significant digits:

| Decimal notation | Scientific notation | Significant digits |
|---|---|---|
| 123456789012 | 1.23456789012E11 | 12 |
| .0000123456 | 1.23456E-05 | 6 |
| -12300000000 | -1.23E10 | 3 |

NOTE

A large number, such as -12300000000, can have a small number of significant digits, and a small number, such as .0000123456, can have a large number of significant digits.

The previous table also shows when MANTIS displays a number in scientific notation rather than decimal notation when processing a SHOW statement. MANTIS does this if the number is:

- Greater than or equal to 1E11 (100,000,000,000).
- Less than 1E-4 (.0001) but greater than -1E-4 (-.0001).
- Less than or equal to -1E10 (-10,000,000,000).

You can enter any number in scientific notation. In this case, the number preceding the E can have more than one digit before the decimal point. For example, you can enter:

- `100` as `1E2`
- `123000` as `123E3`
- `1.7640` as `17640E-4.`
- `.0004` as `4E-4`

## When to use BIG and SMALL

MANTIS stores numbers in precision of:

- **BIG.** Holds up to 15 significant digits.
- **SMALL.** Holds up to 6 significant digits.

BIG variables provide for much greater accuracy in your calculations than SMALL variables. Use BIG variables unless you are concerned with the storage requirement of your program (BIG variables require approximately twice as much space as SMALL variables). Do not use SMALL for decimals. If you do not define a variable as BIG or SMALL, MANTIS assumes BIG.

MANTIS accepts any number in the range 1E-38 through 1E+38 in SMALL precision and 1E-308 through 1E+308 in BIG precision. A program fault will occur if a calculation results in a number outside these ranges. MANTIS performs all calculations internally using BIG precision.

Consider the following program:

```
10 SMALL ALPHA
20 ALPHA = 987E7
```

In statement 10, ALPHA is a variable with a SMALL precision. Statement 20 assigns a value of 9870000000 to ALPHA. Since it has only three significant digits, it can be stored without loss of accuracy in the significant digits, although conversion to internal floating-point format and back may introduce a slight rounding error. In this case, MANTIS will return the value of ALPHA as 9870000128 after conversion and storage.

If, however, you specify the second line as:

```
20 ALPHA = 9876543210
```

which has nine significant digits, there will be a loss of accuracy in the significant digits since MANTIS can accommodate only six significant digits for a SMALL variable. After conversion and storage, MANTIS will return the value of ALPHA as 9876543488. Only the first seven significant digits are correct.

If you specify ALPHA as:

```
10 BIG ALPHA
```

you will not lose accuracy in the significant digits since a BIG specification holds up to 15 significant digits. The rounding errors that may be incurred during conversion or calculation are much smaller with BIG variables than with SMALL variables.

MANTIS handles numeric precision on a display (with LIST or SHOW) in a different way. For example, if you enter the following:

```
10 BIG A,B
20 A=1.000000000002
30 B=9.999999999e11
```

the result when you use the LIST command is:

```
10 BIG A,B
20 A=1
30 B=1e12
```

The numeric literal value of the variables exceeds the precision of MANTIS numeric printing capabilities provided by MANTIS. In addition, the following is possible:

```
10 A=120000000.0002      lists as      10 A=120000000
20 B=120000000.0001      lists as      20 B=120000000
```

If you run SHOW (A=B) after lines 10 and 20 above, the result is 0. Even though A and B appear to be equal when you use the LIST command, they are not equal according to internal numeric precision. That is, the numbers do not exceed numeric comparison precision in MANTIS, but do exceed the precision of numeric printing.

To avoid confusion, the number of digits in a numeric literal should be limited to the number of displayable digits. If it is necessary to display more than the number of displayable digits, you can use the FORMAT function to format the numeric value according to a display mask with more than the number of displayable digits (see "FORMAT" on page 136). The resulting text value can then be displayed using the SHOW statement.

## Arithmetic expressions

An arithmetic expression in MANTIS consists of one or more numeric operands.  The following table lists the various types of numeric operands with an example of each type:

| Operand | Example |
|---|---|
| Numeric variable | ALPHA, BETA, GAMMA |
| Numeric array element | A(5), C(2), X(Y+Z) |
| Numeric constant | 1.43, 0.07, 14E-7 |
| Built-in numeric function | SIN(Y), NOT(Z) |
| Built-in numeric constant | PI, E, FALSE |
| Subexpression in parentheses | (4+SIN(Y)) |

The simplest type of expression is an operand by itself (e.g., ALPHA).  In expressions with more than one operand, each pair of operands is separated by an operator (e.g., ALPHA+BETA-GAMMA).  An operator is a symbol for an arithmetic operation (e.g., + is the symbol for addition). The following table lists the arithmetic operators with a brief description of their operations and an example of each one:

| Symbol | Operation | Example |
|---|---|---|
| ** | Raise A to the power B. | A**B |
| * | Multiply A by B. | A*B |
| / | Divide A by B. | A/B |
| + | Add A to B. | A+B |
| - | Subtract B from A. | A-B |

NOTE

If you have the expression A/B, and B is equal to zero (normally an infinitely large number), MANTIS sets the value of the expression to zero to avoid error conditions.

The first operand in an expression can be preceded by a + or - operator, which is then called a unary operator (e.g., -ALPHA+BETA). The unary operator - changes the sign of the first operand. The unary operator + has no effect. To change the sign of an operand other than the first, enclose it in parentheses so that it becomes the first operand of a subexpression (e.g., ALPHA/(-GAMMA)).

MANTIS evaluates an arithmetic expression by applying the arithmetic operators to the operands in the following order:

```
Unary + -
**
*,/
+,-
```

MANTIS evaluates a subexpression enclosed in parentheses before the expression in which it is an operand. For example, if you enter:

```
YEAR+DAY/(6*(MONTH-LAG))
```

MANTIS subtracts LAG from MONTH, multiplies the result by 6, divides DAY by the product and adds YEAR to the result of the division.

When two operators have equal priority (e.g., + -), MANTIS evaluates the expression from left to right. When evaluating expressions, MANTIS converts all numbers into BIG format and keeps all internal intermediate results in BIG format.

## Relational and logical expressions

A relational expression is an expression that evaluates to a value of true or false, depending on the relationship between its operands. MANTIS has no special data type for the logic values TRUE and FALSE, that are treated as numeric values. All nonzero values represent TRUE; zero represents FALSE. Relational and logical operators compare two numeric operands or two text operands and produce a result of 1 (TRUE) or 0 (FALSE). Relational operators are shown in the following table:

| Symbol | Operation | Example |
|---|---|---|
| = | If A equals B, the value of A=B is TRUE; otherwise, the value is FALSE. | A = B |
| < | If A is less than B, the value of A<B is TRUE; otherwise, the value is FALSE. | A < B |
| > | If A is greater than B, the value of A>B is TRUE; otherwise, the value is FALSE. | A > B |
| <> | If A does not equal B, the value of A<>B is TRUE; otherwise, the value is FALSE. | A <> B |
| >= | If A is greater than or equal to B, the value of A>=B is TRUE; otherwise, the value is FALSE. | A >= B |
| <= | If A is less than or equal to B, the value of A<=B is TRUE; otherwise the value is FALSE. | A <= B |

When comparing text operands, MANTIS first compares their lengths and compares their values only if the lengths are equal. For example, the value of “MANHATTAN” > “NEW YORK” is TRUE because the SIZE of MANHATTAN is nine characters and that of NEW YORK is eight characters.

If the text operands are equal in length, MANTIS compares character-by-character from left to right using the standard ASCII collating sequence. For example, the value of “ANNE” > “ANNA” is TRUE because “E” is higher in the collating sequence than “A”.

Logical operators combine two logic values and produce a result of TRUE or FALSE. Logical operators are shown in the following table:

| **Symbol** | **Operation** | **Example** |
|---|---|---|
| AND | If A is true and B is true, the value of A AND B is TRUE; otherwise, the value is FALSE. | A AND B |
| OR | If A is true or B is true, the value of A OR B is TRUE; otherwise, the value is FALSE. | A OR B |

When the relational operands are numeric, some of the least significant digit-bits are truncated internally prior to making the comparison. This compensates for lost accuracy associated with floating-point arithmetic and conversion to floating-point. The number of significant digits in a numeric comparison is approximately 15.

You can mix operators of all types (arithmetic, text, relational, and logical) in a relational, logical, or numeric expression (e.g., A < B + C or A > D * E). MANTIS evaluates operators in the following order:

```
&
Unary + -
**
* /
+ -
=, >, <, > =, < =, < >
AND
OR
```

When the operators have equal priority, MANTIS evaluates the expression from left to right. Subexpressions within parentheses are evaluated first.

The equivalence of numeric and logical values allows an arithmetic expression to be used in place of a relational expression and vice versa, although the result may not be meaningful.  While accepting any nonzero value as TRUE, MANTIS always evaluates a TRUE result to 1.  FALSE always evaluates to 0.

NOTE

Both terms of AND/OR are evaluated, regardless of the value of the first term.  For example, in the expression IF A AND B, both A and B are evaluated even if A is FALSE.  Therefore, IF I > 0 AND X(I) = 3 will fail if X(I) <= 0 since X(I) will be evaluated.

You can use a logical expression to simplify code that otherwise needs IF or WHEN constructions.  For example, to determine payment terms, you can replace:

```
IF CREDIT RATING="A"
.TERMS=30
ELSE
.IF CREDIT RATING="B"
..TERMS=10
.ELSE
..TERMS=0
.END
END
```

with

```
TERMS=((CREDIT RATING="B")*10)+((CREDIT RATING="A")*30)
```

## Indirect reference (&)

You can use an ampersand (&) to make an indirect reference to a symbolic name. You can use indirect reference for any symbolic name except for the name and parameters of an ENTRY statement. The ampersand acts on the next term of an expression, that must result in a text string, and takes precedence over all other operators and subscripts. The following examples show indirect reference:

| Code with redirection equivalent | Code |
|---|---|
| `TEXT T1,T2: T1="X": T2="Y"` | `TEXT T1,T2: T1="X": T2="Y"` |
| `SMALL &T1` | `SMALL X` |
| `BIG &(T1+T1)` | `BIG XX` |
| `TEXT &T2(4,10)` | `TEXT Y(4,10)` |
| `&T2(3)=T1` | `Y(3)=T1:\| T1 contains "X"` |
| `&(&T2(3))=123` | `&(Y(3))=123` |
| which further resolves to: | `X=123` |

NOTE

Use caution when using indirect reference through an array. To ensure that the array reference is evaluated first, it must be enclosed in parentheses. The following example causes Y(4) to be evaluated before the indirect reference operator is applied. If the indirect reference is to an array, any subscripts for that array must follow the indirect reference:

```
&(Y(4))
```

The following example illustrates this point:

```
TEXT Y(8,32), X1 (10, 16)
Y(4) = "x1"
&(Y(4)) (3)= "customer"
```

## Built-in functions

Many numeric and text functions are built into MANTIS. Functions that do not have any arguments are sometimes referred to as built-in constants or built-in variables. You can use subscripts to obtain a substring of a built-in text variable. All functions are described in detail in "Statements and built-in functions," beginning on page 65.

# 3

# MANTIS programming language

## Statements and built-in functions

In addition to describing each statement and built-in function, this chapter provides the following information for each parameter of the statement or function:

- ♦ **Description**. Description of the parameter.
- ♦ **Default**. Default value, if any, of the parameter.
- ♦ **Format**. Required format of the parameter.
- ♦ **Options**. Values you can supply for the parameter.
- ♦ **Example**. Example of the statement or function.
- ♦ **Considerations**. Any special limitations, considerations, and guidelines for the statement or function.

NOTE

Some statements, functions, and parameters can be used on MANTIS for Windows, but are not currently supported by MANTIS for the IBM mainframe. These statements, functions, and parameters are noted as such in their first consideration. "Compatibility considerations (personal computer and IBM mainframe)," beginning on page 283, discusses compatibility between the two environments.

The following table outlines the MANTIS statements and functions that are ordered alphabetically in this chapter. Also included in this table is a brief description of each statement or function and the page where you can find more information.

| **Statement/Function** | **Enables you to. . .** | **See** |
|---|---|---|
| ABS(*a*) | Return the absolute value of an arithmetic expression. MANTIS evaluates the expression and changes the sign of the result if it is negative. | "ABS" on page 74. |
| ACCESS | Define an external file your program will access. | "ACCESS" on page 75. |
| ATN(*a*) | Return the angle in radians whose tangent is an arithmetic expression. | "ATN" on page 80. |
| ATTRIBUTE | Change the attributes associated with a screen and its fields. | "ATTRIBUTE" on page 81. |
| ATTRIBUTE (PRINTER) | Return a text value indicating the current printer attributes. | "ATTRIBUTE (PRINTER)" on page 87. |
| BIG | Define numeric variables and arrays with up to 15 significant digits. | "BIG" on page 88. |
| BREAK* | Exit from FOR-END, UNTIL-END, WHEN-END, and WHILE-END statements. | "BREAK" on page 89. |
| CALL | Call an interface program. | "CALL" on page 90. |
| CHAIN | Replace the program currently in the work area with another program and begin executing that program. | "CHAIN" on page 93. |

* This statement/function is not supported by Releases 5.2 and above of MANTIS for the IBM mainframe.

| Statement/Function | Enables you to. . . | See |
|---|---|---|
| CHR(*a*,...)* | Return a text value consisting of the character(s) corresponding to the ASCII code(s) specified. | "CHR" on page 95. |
| CLEAR | Clear the screen, data, or the fault trap. | "CLEAR" on page 96. |
| COMMIT | Flush any updated buffers or enable/disable automatic COMMIT. | "COMMIT" on page 98. |
| CONVERSE | Send a formatted screen design to a terminal and return the responses made by the operator. | "CONVERSE" on page 100. |
| COS(*a*) | Return the cosine of an arithmetic expression. | "COS" on page 107. |
| CURSOR(...) | Indicate whether the cursor was located in a specific field at the completion of the last terminal I/O. | "CURSOR" on page 108. |
| DATAFREE | Return the number of free bytes in the data work area. Provided for compatibility with MANTIS for the IBM mainframe. | "DATAFREE" on page 110. |
| DATE | Function: Return the text value of the current date.<br><br>Statement: Specifies the date mask to be used by the DATE function. | "DATE" on page 111. |
| DELETE | Delete a record from a file or a view. | "DELETE" on page 114. |
| DEQUEUE | Provided for compatibility with MANTIS for the IBM mainframe. MANTIS for Windows ignores this statement. | "DEQUEUE" on page 118. |
| DO | Execute an internal or external subroutine. | "DO" on page 119. |
| DOLEVEL | Return the current level in an external subroutine. | "DOLEVEL" on page 122. |
| E | Return the value of e (2.71828182846). | "E" on page 123. |

* This statement/function is not supported by Releases 5.2 and above of MANTIS for the IBM mainframe.

| Statement/Function | Enables you to. . . | See |
|---|---|---|
| EDIT* | Invoke an external text editor to edit a text variable or array. | "EDIT" on page 124. |
| ENQUEUE | Provided for compatibility with MANTIS for the IBM mainframe. MANTIS for Windows ignores this statement. | "ENQUEUE" on page 125. |
| ENTRY-EXIT | Define the boundary of a subroutine or program. | "ENTRY-EXIT" on page 126. |
| EXP(*a*) | Return the value of e to the power of an arithmetic expression. | "EXP" on page 128. |
| FALSE | Return the value FALSE (zero). | "FALSE" on page 129. |
| FILE | Define a MANTIS file your program will access. | "FILE" on page 130. |
| FOR-END* | Repeat execution of a block of statements while a counter is incremented or decremented through a range of values. | "FOR-END" on page 134. |
| FORMAT(*a*,*t*) | Return a text value resulting from the conversion of a numeric expression according to the supplied edit mask. | "FORMAT" on page 136. |
| FSI(...) | Return additional status for an external file following a GET/UPDATE/INSERT/DELETE. | "FSI" on page 137. |
| GET | Read a record from a file or view. | "GET" on page 139. |
| HEAD | Specify a heading to be centered on the top line of the screen in scroll mode. | "HEAD" on page 147. |

* This statement/function is not supported by Releases 5.2 and above of MANTIS for the IBM mainframe.

| **Statement/Function** | **Enables you to. . .** | **See** |
|---|---|---|
| IF-ELSE-END | Execute one block of statements if a specified condition is TRUE, another block if the condition is FALSE. | "IF-ELSE-END" on page 149. |
| INSERT | Insert a new record into a file or view. | "INSERT" on page 151. |
| INT(*a*) | Return the integer value of an arithmetic expression, truncating fractions toward zero. | "INT" on page 155. |
| INTERFACE | Define an interface that your program will access. | "INTERFACE" on page 156. |
| KEY | Return a text value identifying your response to the most recent CONVERSE, OBTAIN, PROMPT, or WAIT statement. For CONVERSE, the value is the name of the action or macro key you entered (see the table in the "MANTIS macro keys" section on page 36). For OBTAIN, PROMPT, and WAIT, this value is replaced by the contents of the Key Simulation field. | "KEY" on page 159. |
| LANGUAGE* | Return the default language of the user who is currently signed on. | "LANGUAGE" on page 160. |
| LET | Assign a value to a variable or array element. | "LET" on page 161. |
| LOG(*a*) | Return the natural logarithm of an arithmetic expression. | "LOG" on page 164. |
| MODIFIED(...) | Return the number of fields in a screen that was modified during the most recent CONVERSE, or indicate whether a specific field was modified. | "MODIFIED" on page 165. |
| NEXT* | Execute the next repeat in FOR-END, UNTIL-END, or WHILE-END statements or the next condition in WHEN-END statements. | "NEXT" on page 167. |
| NOT(*a*) | Return TRUE if an arithmetic expression evaluates to FALSE; otherwise, return FALSE. | "NOT" on page 168. |

* This statement/function is not supported by Releases 5.2 and above of MANTIS for the IBM mainframe.

| Statement/Function | Enables you to. . . | See |
|---|---|---|
| NULL* | Return a null (zero-length) text value. | "NULL" on page 168. |
| NUMERIC* | Determine if a text expression contains a valid number. | "NUMERIC" on page 169. |
| OBTAIN | Obtain input from the scroll-mode screen and assign input data to variables. | "OBTAIN" on page 170. |
| OUTPUT | Route output from CONVERSE or SHOW statements to the screen, the printer, or both. | "OUTPUT" on page 172. |
| PAD | Insert padding characters into a text value. | "PAD" on page 174. |
| PASSWORD | Return a text value containing the password entered during MANTIS sign-on. | "PASSWORD" on page 177. |
| PERFORM | Invoke a DOS command. | "PERFORM" on page 178. |
| PI | Return the value of pi (3.14159265359). | "PI" on page 180. |
| POINT(*t*±*t*) | Return the character position where joining or removal occurs in a text expression. | "POINT" on page 181. |
| PRINTER | Return a text value containing the current printer assignment or assign a new value. | "PRINTER" on page 182. |
| PRINTER= | Directs MANTIS printer output to a file or device. | "PRINTER=" on page 183. |
| PROGFREE | Return the number of free bytes in the program work area. | "PROGFREE" on page 186. |
| PROGRAM | Define an external subroutine for your program to use. | "PROGRAM" on page 187. |
| PROMPT | Display a prompter. | "PROMPT" on page 189. |
| RELEASE (statement) | Close an external file or interface, or release storage used by an external subroutine. | "RELEASE" on page 190. |

*	This statement/function is not supported by Releases 5.2 and above of MANTIS for the IBM mainframe.

| Statement/Function | Enables you to. . . | See |
|---|---|---|
| RELEASE(function) | Obtain the current release and version number of MANTIS and the operating environment. | "RELEASE" on page 193. |
| RESET | Back out a Logical Unit of Work (LUW). | "RESET" on page 194. |
| RETURN* | Return control from a subroutine. | "RETURN" on page 195. |
| RND(*a*) | Return a random real number in the range zero to a, excluding zero and a. | "RND" on page 196. |
| SCREEN | Define a screen design your program will use. | "SCREEN" on page 197. |
| SCROLL | Set the amount by which windowing keys will move the physical screen on the logical display. | "SCROLL" on page 199. |
| SEED | Direct the random number generator to generate a new sequence of random numbers. | "SEED" on page 201. |
| SET* | Set a fault trap or assign a value to a system environment variable. | "SET" on page 202. |
| SGN(*a*) | Return -1 if an arithmetic expression is less than 0, 0 if it equals 0, and +1 if it is greater than 0. | "SGN" on page 205. |
| SHOW | Display data in scroll mode or display the name of the fault trap routine. | "SHOW" on page 206. |
| SIN(*a*) | Return the sine of an arithmetic expression in radians. | "SIN" on page 209. |
| SIZE(...) | Return the size of a text value or the size and dimensions of a field. | "SIZE" on page 210. |
| SLICE | Specify the number of statements in a program slice. | "SLICE" on page 212. |
| SLOT | Specify the number of program slices to be executed before a loop warning. | "SLOT" on page 214. |

* This statement/function is not supported by Releases 5.2 and above of MANTIS for the IBM mainframe.

| Statement/Function | Enables you to. . . | See |
|---|---|---|
| SMALL | Define numeric variables and arrays with a maximum of six significant digits. | "SMALL" on page 215. |
| SQR(*a*) | Return the square root of an arithmetic expression. | "SQR" on page 216. |
| STOP | Stop program execution. | "STOP" on page 217. |
| $SYMBOL(*t*)* | Return the text value assigned to a system environment variable. | "$SYMBOL" on page 218. |
| TAN(*a*) | Return the tangent of an arithmetic expression in radians. | "TAN" on page 219. |
| TERMINAL | Return a text value containing the terminal ID.  For compatibility with MANTIS for the IBM mainframe. | "TERMINAL" on page 220. |
| TERMSIZE | Return a text value giving the size of the physical screen in rows and columns. | "TERMSIZE" on page 221. |
| TEXT | Define text variables and arrays. | "TEXT" on page 222. |
| TIME | Function:  Return a text value expressing the current time.<br>Statement:  Specifies the time mask to be used by the TIME function. | "TIME" on page 224. |
| TRAP | Intercept certain error conditions during access to a MANTIS file, external PC file, or interface program. | "TRAP" on page 228. |
| TRUE | Return the value TRUE (1). | "TRUE" on page 231. |
| TXT(*a*) | Return the text value of an arithmetic expression in the default numeric display format. | "TXT" on page 232. |
| UNPAD | Remove padding characters from a text value. | "UNPAD" on page 233. |

*	This statement/function is not supported by Releases 5.2 and above of MANTIS for the IBM mainframe.

| **Statement/Function** | **Enables you to. . .** | **See** |
|---|---|---|
| UNTIL-END | Repeat execution of a block of statements until a specified condition becomes TRUE. | “UNTIL-END” on page 236. |
| UPDATE | Replace a record on a file or view with an updated record. | “UPDATE” on page 237. |
| UPPERCASE(*t*)* | Return the text value of a text expression with any lowercase letters changed to uppercase. | “UPPERCASE” on page 241. |
| USER | Return a text value containing the user name entered during MANTIS sign-on. | “USER” on page 242. |
| USERWORDS | Return the number of MANTIS symbolic names in use. | “USERWORDS” on page 243. |
| VALUE(*t*) | Return the numeric value of a text expression.  MANTIS ignores all nonnumeric characters except for decimal point (.), minus sign (-), and exponent indicator (E). | “VALUE” on page 244. |
| WAIT | Wait for a terminal action or macro key to be pressed. | “WAIT” on page 245. |
| WHEN-END | Execute a block of statements when a specified condition is TRUE. | “WHEN-END” on page 246. |
| WHILE-END | Repeat execution of a block of statements while a specified condition is TRUE. | “WHILE-END” on page 248. |
| ZERO | Return the value zero. | “ZERO” on page 250. |

*	This statement/function is not supported by Releases 5.2 and above of MANTIS for the IBM mainframe.

## ABS

Use the ABS function to return the absolute value of an arithmetic expression.

**ABS(*a*)**

***a***

**Description** *Required.* Specifies the expression whose absolute value you want returned.

**Format** Valid arithmetic expression.

**Examples**

| | | |
|---|---|---|
| `ABS(0)` | returns | `0` |
| `ABS(-14E9)` | returns | `14E9` |
| `ABS(-.5)` | returns | `.5` |

## ACCESS

Use the ACCESS statement to define an external file view that your program will access. MANTIS retrieves the external file view from the user library and defines variables in the work area for each field in the view.

**ACCESS *access-name*(*libname*,*password*[,PREFIX][,*levels*]**

**[,NEW | ,REPLACE][,FILE *extname*]),...**

### *access-name*

**Description** *Required.* Provides the symbolic name of the external file view for use in subsequent GET, UPDATE, INSERT, DELETE, RELEASE, and TRAP statements.

**Format** Unique MANTIS symbolic name.

**Considerations**

- No processing occurs if *access-name* is already defined.
- The *access-name* symbolic name returns a text value indicating the status of the most recent GET, UPDATE, INSERT, or DELETE operation performed on an external file view.
- The physical open of the file does not take place until the first GET, UPDATE, INSERT, or DELETE statement is executed. You may use the TRAP statement to trap file open errors.

**Example**

```
20 .ACCESS RECORD("INDEX","SERENDIPITY",16)
30 .SCREEN MAP("INDEX")
40 .WHILE RECORD<>"END" AND MAP<>"CANCEL"
50 ..CLEAR MAP:LEVEL NUMBER=1
60 ..GET RECORD LEVEL=LEVEL NUMBER
```

### *libname*

**Description** *Required.* Specifies the library name of the external file view, as specified during External File View Design.

**Format** Text expression that evaluates to [*user name*:]*external-file-view-name*.

**Consideration** If the external file view is in your own library, you do not need to specify user name.

### *password*

**Description** *Required.* Specifies the password needed for the type of access (GET, UPDATE, INSERT/DELETE) your program requires.

**Format** Text expression that evaluates to the password specified during External File View Design.

**Consideration** The specified password determines the file access mode used to open the file.  The password for altering and the password for deleting/inserting set write access.  The password for viewing sets read access.

### PREFIX

**Description** *Optional.* Put the prefix *access-name* on all variable names associated with this external file view.  For example, if the external file view BOTTLES has a field named VOLUME, the following statement causes the corresponding variable BIN VOLUME to be defined in the work area:

```
10 ACCESS BIN("BOTTLES","MAGIC",PREFIX)
```

**Consideration** MANTIS also prefixes the reference variable specified in the external file view for a SEQUENTIAL or RELATIVE file.

## *levels*

**Description** *Optional.* Specifies the number of levels you want to use in GET, UPDATE, INSERT, or DELETE statements for the external file.

**Format** Arithmetic expression that evaluates to a value in the range 1–255.

**Considerations**

- ♦ If you do not specify *levels*, MANTIS defines a BIG, SMALL, or TEXT variable or array for each field in the external file view according to the type and dimensions specified during External File View Design.
- ♦ If you specify *levels*, MANTIS adds an extra dimension to each field to allow a separate value to be stored at each level. MANTIS defines a one-dimensional array instead of a variable, a two-dimensional array instead of a one-dimensional array, and so on. The first dimension of each array is levels.

  For example, a field X defined as a 4-byte float field with dimensions of 10 and 20 is equivalent to SMALL X(10,20). If the ACCESS statement specifies *levels* as 16, then it is equivalent to SMALL X(16,10,20).
- ♦ If you specify *levels*, you must also specify LEVEL=*level-number* (unless you want the default LEVEL=1) in all GET, UPDATE, INSERT, and DELETE statements for the external file view.

## NEW

**Description** *Optional.* Creates a new version of the file (primary key) or creates a secondary index (alternate key).

**Considerations**

- All
  - This parameter is not supported by MANTIS for the IBM mainframe.
  - If you specify NEW, you must use the delete/insert password.
- Primary key
  - If the file already exists, the file open will fail.
  - If you do not specify NEW or REPLACE, the external file must already exist or the file open will fail.
- Alternate key
  - If the file does not exist, the file open will fail.
  - If the key of reference specified in the file view already exists, the file open will fail.
  - If you do not specify NEW, the alternate key must already exist or the file open will fail.
  - If the file already contains records, alternate keys will be created for those records. If this results in any error (e.g., duplicate keys), the file open will fail and the alternate index will not be created.
  - You may have up to 15 alternate indexes for a file.

## REPLACE

**Description** *Optional.* Replace the old version of the external file. If the file does not exist, it will be created.

**Considerations**

- ♦ This parameter is not supported by MANTIS for the IBM mainframe.
- ♦ If you specify REPLACE, you must use the delete/insert password.
- ♦ If you do not specify REPLACE or NEW, the external file must already exist or the ACCESS statement will fail.

## FILE *extname*

**Description** *Optional.* Specifies an external file name that is used instead of the name provided in the external file view.

**Format** Text expression that evaluates to a valid PC file name.

**Consideration** This parameter is not supported by MANTIS for the IBM mainframe.

**Example**

```
20 .ACCESS RECORD("INDEX","SERENDIPITY",16)
30 .SCREEN MAP("INDEX")
40 .WHILE RECORD"CANCEL"
50 ..CLEAR MAP:LEVEL NUMBER=1
60 ..GET RECORD LEVEL=LEVEL NUMBER
```

## ATN

Use the ATN (arctangent) function to return the angle in radians whose tangent is an arithmetic expression.

**ATN(*a*)**

***a***

**Description** *Required.* Specifies the arithmetic expression whose arctangent you want returned.

**Format** Valid arithmetic expression.

**Examples**

| | | |
|---|---|---|
| `ATN(10)` | returns | `1.4711276743` |
| `ATN(100)` | returns | `1.56079666011` |

## ATTRIBUTE

Use the ATTRIBUTE statement to change the attributes of a field, a screen, or a printer. The changed attributes remain in effect until you change them again or use the RESET attribute to revert to the original specification.

$$\textbf{ATTRIBUTE} \begin{Bmatrix} (\textbf{\textit{screen - name}}\ [\textbf{,}\ \textbf{\textit{field - name}}]) \\ (\textbf{PRINTER}) \end{Bmatrix} \ \textbf{=}\ \textbf{\textit{attributes}}\textbf{,}\ldots$$

### *screen-name*

| | |
|---|---|
| **Description** | *Optional.* Specifies the symbolic name of a screen design, as defined in a SCREEN statement. |
| **Format** | MANTIS symbolic name defined in a previously executed SCREEN statement. |

### *field-name*

| | |
|---|---|
| **Description** | *Optional.* Specifies the field whose attributes you want to change. |
| **Format** | Symbolic name of a field in the screen, as specified in Screen Design. If the field is repeated in the screen, *field-name* must be subscripted to specify the array element to which the ATTRIBUTE statement applies. |
| **Consideration** | If you omit *field-name*, the attributes apply to all fields in the screen or to the screen itself, depending upon the attributes you specify. |

### PRINTER

| | |
|---|---|
| **Description** | *Optional.* Changes the attributes of the printer. |
| **Consideration** | In MANTIS for the IBM mainframe, the PRINTER parameter of the ATTRIBUTE statement is restricted to the Master User. |

## ***attributes***

**Description** *Required.* Specifies one of the following different levels of attributes:

- **Field.** Apply to a field on a screen.
- **Map.** Apply to the screen.
- **Device.** Apply to the printer device .

Refer to the *MANTIS for Windows Facilities Reference Manual*, P19-2301, for information about Field-level and map-level attributes. Refer to the *MANTIS for Windows Administration Guide*, P19-2304, for information about device-level attributes.

**Format** Each attribute's parameter must be a text expression that evaluates to one or more attributes (minimum 3-character abbreviation for each attribute, as listed under Options), with each attribute separated by a comma (e.g., CUR,PRO).

**Options** Field-level attributes, paired by opposites, are:

| | | |
|---|---|---|
| AUTOSKIP | NO AUTOSKIP | |
| BLINK | NO BLINK | |
| BRIGHT | NORMAL | HIDDEN |
| CURSOR | | |
| DETECTABLE* | NON DETECTABLE* | |
| HIGHLIGHT | NO HIGHLIGHT | |
| LEFT BAR* | NO LEFT BAR* | |
| MODIFIED* | UNMODIFIED* | |
| NO COLOR | BLUE, GREEN, NEUTRAL, PINK, RED, TURQUOISE, YELLOW | |
| OVERLINE* | NO OVERLINE* | |
| PROTECTED | UNPROTECTED | |
| RESET | | |
| REVERSE VIDEO | VIDEO | |
| RIGHT BAR* | NO RIGHT BAR* | |
| UNDERLINE | NO UNDERLINE | |
| UPPERCASE | LOWERCASE | |

Map-level attributes, paired by opposites, are:

| | |
|---|---|
| ALARM | NO ALARM |
| PROTECTED BOTTOM LINE | BOTTOM LINE UNPROTECTED |
| FULL SCREEN DISPLAY | NOT FULL SCREEN DISPLAY |
| WINDOWING* | NO WINDOWING* |

* This attribute has no effect in MANTIS for Windows. It is included for compatibility with MANTIS for the IBM mainframe.

Device-level attributes, paired by opposites, are:

| | |
|---|---|
| SPOOL ON CLOSE | NO SPOOL ON CLOSE |
| CLASS | |
| BLINK | NO BLINK |
| BRIGHT | NORMAL |
| COLOR | NO COLOR |
| DETECTABLE* | NON DETECTABLE* |
| REVERSE VIDEO | VIDEO |
| UNDERLINE | NO UNDERLINE |

* This attribute has no effect in MANTIS for Windows. It is included for compatibility with MANTIS for the IBM mainframe.

## Considerations

### Field-level attributes

- If you do not specify *field-name*, field-level attributes (except CURSOR) apply to all fields in the specified screen.
- MANTIS ignores any characters entered after the first three. For example, you may specify BRIGHT as “BRI”, “BRIGHT”, or “BRILLIG”.
- Each attribute that you specify resets any opposite attribute that is in effect. In addition, BRIGHT sets DETECTABLE and resets HIDDEN; NORMAL resets BRIGHT and HIDDEN; HIDDEN resets BRIGHT and DETECTABLE; DETECTABLE resets HIDDEN; and CURSOR resets CURSOR on all other fields in the screen.
- The HIGHLIGHT attribute causes the field to be displayed with any highlighting attribute (BRIGHT or UNDERLINE) that the output device supports. If both BRIGHT and UNDERLINE are supported, BRIGHT is used for screen output and UNDERLINE is used for printer output.
- The RESET attribute resets all attributes of the screen field to the original settings specified in Screen Design.
- You cannot change the attributes of heading fields by using the ATTRIBUTE statement. (To change these attributes, you must use the Update Field Specifications option in Screen Design, as documented in *MANTIS for Windows Facilities Reference Manual*, P19-2301.)

### Device-level attributes

- Specifies the CLASS attribute with a *device-name* in parentheses, for example, "CLASS(*device-name*)".  The *device-name* refers to a printer definition that is stored in an external PC file, PRINTER.DEF. Definitions in this file can be modified, inserted, or deleted through the Update Printer Definitions Facility (refer to the *MANTIS for Windows Administration Guide*, P19-2304).  MANTIS processes a CLASS attribute by obtaining device attributes from the specified printer definition in the PRINTER.DEF file.  Printer definitions may be set up for nonstandard printers at your installation.

- By default, MANTIS supports BRIGHT and UNDERLINE in printed screens and simulates these attributes if your printer does not support them.  Specify NORMAL or NO UNDERLINE if you want MANTIS to ignore these attributes.

- You can specify the size of a printed page in rows and columns as in the following example:

```
ATTRIBUTE(PRINTER)="(60,132)"
```

  MANTIS selects the most appropriate printer mode defined for the current printer CLASS.  (Refer to the *MANTIS for Windows Administration Guide*, P19-2304.)

## Examples

♦ The following example sets the ACCT NUM field associated with the screen INVOICE to BRIGHT and PROTECTED:

```
10 SCREEN INVOICE("PROGRAMA")
20 TEXT STANDARD(16),TEMP(16)
30 STANDARD="BRIGHT,PROTECTED"
40 ATTRIBUTE(INVOICE,ACCT NUM)=STANDARD
```

♦ The following example produces the same result as the previous example:

```
30 STANDARD="BRIGHT"
40 ATTRIBUTE(INVOICE,ACCT NUM)=STANDARD+
50 "',PROTECTED"
```

♦ The following example sets the ACCT NUM field associated with the screen INVOICE to BRIGHT, PROTECTED, and CURSOR:

```
30 STANDARD="PROTECTED"
40 TEMP="CURSOR"
50 ATTRIBUTE(INVOICE,ACCT NUM)="BRIGHT"
.
.
.
160 ATTRIBUTE(INVOICE,ACCT NUM)=STANDARD
.
.
.
240 ATTRIBUTE(INVOICE,ACCT NUM)=TEMP
```

♦ The following example produces the same result as the previous example:

```
240 ATTRIBUTE(INVOICE,ACCT NUM)="PRO,BRI,CUR"
```

♦ The following example resets all attributes to their original specifications:

```
240 ATTRIBUTE(INVOICE)="RESET"
```

## ATTRIBUTE (PRINTER)

Use the ATTRIBUTE(PRINTER) function to return the attributes of the printer.

**ATTRIBUTE(PRINTER)**

### PRINTER

**Description** *Required.* Returns the attributes of the printer.

**Format** Printer attributes paired by opposites are:

| | |
|---|---|
| SPOOL ON CLOSE | NO SPOOL ON CLOSE |
| CLASS | |
| BRIGHT | NORMAL |
| UNDERLINE | NO UNDERLINE |

**Example**

```
SHOW ATTRIBUTE(PRINTER)
```

### General consideration

In MANTIS for the IBM mainframe, the ATTRIBUTE function is restricted to the Master User.

## BIG

Use the BIG statement to define numeric variables or arrays of numeric variables to hold values with a maximum of 15 significant digits. Values are initially set to zero.

**BIG *big-name*[(*dimension*,...)],...**

### *big-name*

| | |
|---|---|
| **Description** | *Required.* Provides the symbolic name of the numeric variable or array. |
| **Format** | MANTIS symbolic name. |
| **Consideration** | No processing occurs if *big-name* is already defined. |

### *dimension*

| | |
|---|---|
| **Description** | *Optional.* Specifies an array dimension. Use one dimension parameter to specify a one-dimensional array, two dimension parameters to specify a two-dimensional array, and so on. |
| **Format** | Arithmetic expression that evaluates to a value in the range 1–16000. |
| **Consideration** | You can specify a maximum of 255 dimensions. Each dimension specifies the maximum value for the corresponding array subscript. |

**Example**

```
BIG ALPHA(64,3),BETA
```

**General considerations**

- No processing takes place if *big-name* is already defined.
- The number or size of array dimensions that can be specified and the number of variable names allowed can be reduced in the configuration file (refer to the *MANTIS for Windows Administration Guide*, P19-2304).
- The total size of a single array, including storage management overhead, cannot exceed 64K.
- BIG is the default variable type.

## BREAK

Use the BREAK statement to exit from a FOR-END, UNTIL-END, WHEN-END, or WHILE-END statement. The statement after the END statement is executed next.

**BREAK**

### Example

```
10 FOR L=1 TO MAXLINES
20 .GET CUSTOMER LEVEL=L
30 .IF CUSTOMER="END": ¦      Check status from GET
40 ..BREAK: ¦                 Exit FOR loop if end of file
50 .END
60 END
70 CONVERSE CUST DETAILS
110 WHEN CODE="R"
120 .COLOR="RED"
130 .BREAK
140 WHEN CODE="B"
150 .COLOR="BLUE"
160 .BREAK
170 WHEN CODE="G"
180 .COLOR="GREEN"
190 .BREAK
200 END
```

### General considerations

- ♦ With nested logic statements, BREAK terminates execution of the innermost FOR-END, UNTIL-END, WHEN-END, or WHILE-END block of statements where it occurs.
- ♦ BREAK from a WHEN-END block of statements bypasses any other WHEN conditions in the same block.

## CALL

Use the CALL statement to call an interface program. MANTIS calls the program specified in the interface profile and makes the status returned by the program available.

**CALL *interface-name*[(*argument*,...)][LEVEL=*level-number*]**

### *interface-name*

**Description** *Required.* Specifies the symbolic name of an interface.

**Format** MANTIS symbolic name as defined in a previously executed INTERFACE statement.

### *argument*

**Description** *Optional.* Provides an argument to be passed to the corresponding element in the interface area definition (the value of the first argument is passed to the first element in the interface area definition; the value of the second argument is passed to the second element, etc.).

**Format** Arithmetic or text expression.

**Consideration** When the argument that is passed to an interface is an array, each element of the array must be passed separately.

### LEVEL=*level-number*

**Description** *Optional.* Specifies which level of MANTIS array elements should be passed to/received from the interface.

**Default** LEVEL=1

**Format** Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding INTERFACE statement.

**Example**

```
20 INTERFACE MASTER("CUSTOMERS","ALIBABA",10)
.
.
.
60 CALL MASTER("GET",1234)LEVEL=2
```

## General considerations

- Execution of the CALL statement occurs in three distinct phases:

  1. MANTIS stores the values of the arguments you supply in the corresponding MANTIS variables specified in the interface profile (the first argument is assigned to the first interface field, etc.).

  2. If the DATA FLOW option is specified as OUTPUT or FULL in the interface profile, all of the MANTIS variables in the interface profile are converted to the specified format and stored in the interface area, that is in turn passed to the interface program.

  3. If the DATA FLOW option is specified as INPUT or FULL in the interface profile, the interface area is retrieved from the interface program, and the modified values (in the interface area) are converted back to MANTIS format and stored into the corresponding MANTIS variables.

  Note that when the DATA FLOW option is set to OUTPUT or INPUT, step 3 or step 2, respectively, will not be executed.

- A Dynamic Link Library (DLL) interface operates as a simple subroutine to MANTIS. The DLL module is loaded on the first CALL. The specified routine is called with the address of the interface area.

- The 8-character text status, located at the start of the interface communication area, is set to spaces before the interface is called. This value is returned to MANTIS when the CALL is completed. To obtain the status value, use the *interface-name* built-in text function.

- If TRAP is in effect (see “TRAP” on page 228), CALL may return one of two special statuses (set by MANTIS), TIMEOUT or ERROR.  A TIMEOUT status means that MANTIS was unable to complete the CALL statement in a specified time.  A retry may succeed.  An ERROR status means that MANTIS encountered a serious problem while processing the CALL.  More information can be found with the use of the FSI function that will return one of the following possible values:

| **FSI value** | **Cause** |
|---|---|
| NOTFOUND | A DLL was not found. |
| ROUTINE | A routine was not found in the loaded DLL. |
| MAXPGMS | Too many interface programs were started. |
| MAXDLLS | Too many DLLs were loaded. |

- MANTIS limits the overall number of loaded DLLs to 50.

- MANTIS limits the overall number of interface programs running at the same time to 50.

- For more information, refer to the *MANTIS for Windows Facilities Reference Manual*, P19-2301.

## CHAIN

Use the CHAIN statement to replace the program currently in the work area or at the current DOLEVEL with another MANTIS program and begin executing that program. MANTIS terminates the issuing program and erases all variables after evaluating any arguments being passed.

**CHAIN *libname*[,*argument*,...] [LEVEL]**

### *libname*

**Description** *Required.* Specifies the library name of the program you want to load and execute.

**Format** Text expression that evaluates to [*user name*:]*program-name*.

**Consideration** If the program is in your own library, you need to specify only the *program-name*.

### *argument*

**Description** *Optional.* Specifies the argument(s) to be passed to the new program.

**Format** Either expressions or symbolic names of variables or arrays.

**Considerations**

- ♦ The maximum number of arguments is defined in the configuration file (refer to the *MANTIS for Windows Administration Guide*, P19-2304).
- ♦ The arguments must correspond in number with those specified in the ENTRY statement of the program that you chain to, or all arguments can be omitted from the CHAIN statement. Otherwise, if fewer arguments than necessary are specified, MANTIS supplies extras and sets them to zero.
- ♦ You cannot use a *file-name*, *screen-name*, *access-name*, or *interface-name* as an argument.

## LEVEL

### Description

*Optional.* Indicates that the program at the current DOLEVEL is replaced.

### Example

```
20 HEAD "FACILITY SELECTION"
30 CLEAR
40 SHOW "DESIGN A PROGRAM ....... 1"
50 SHOW "DESIGN A SCREEN ........ 2"
60 SHOW "TERMINATE ............. CANCEL"
70 OBTAIN OPTION
80 WHEN KEY="PF1" OR OPTION=1
90 .CHAIN  "MASTER:PROGRAM"
100 WHEN KEY="PF2" OR OPTION=2
110 .CHAIN "MASTER:SCREEN_DESIGN"
120 WHEN KEY="CANCEL"
132 .CHAIN "MASTER:TERMINATE"
140 END
150 STOP
```

### General considerations

- For compatibility with MANTIS for the IBM mainframe, do not use expressions as arguments.
- Compatibility mode allows a maximum of 40 arguments.
- If your program will execute a CHAIN statement, save the program before you run it in programming mode; otherwise, any changes you have made will be lost when it is replaced by the chained program.
- You can CHAIN to any program in any user’s library.  When the program stops, however, you will not be allowed to list or modify it unless its password is the same as the password of the program it replaced in the work area.
- See “External DO” on page 267 of this manual for additional information on using external DO vs. CHAIN.

## CHR

Use the CHR (character) function to return a text value consisting of the character(s) corresponding to the ASCII code(s) specified.

**CHR(*a*,...)**

***a***

**Description** *Required*. Specifies the ASCII codes whose characters you want returned.

**Format** Valid arithmetic expression in the range 0–255.

**Considerations**

- ♦ MANTIS ignores fractions. For example, CHR(97,97.1,97.9) returns "*aaa*".
- ♦ If the value of the expression is outside the valid range, MANTIS will attempt to use the value MOD 256 (the number is divided by 256, and the remainder is the number used). For example, CHR(97,2**22+97, -(2**22)+97) returns "*aaa*".

**Example**

```
CHR(97,98,99) returns "abc"
```

**General considerations**

- ♦ The CHR function is supported by Releases 5.4 and above of MANTIS for the IBM mainframe.
- ♦ If more than 255 parameters (codes) are specified, MANTIS validates but ignores the additional parameters.

## CLEAR

Use the CLEAR statement to clear the screen, clear the data referred to by a symbolic name, clear all program data, or clear the fault trap.

$$\textbf{CLEAR}\begin{bmatrix}\textbf{\textit{name}}\\ \textbf{ALL}\\ \textbf{TRAP}\end{bmatrix}\textbf{,...}$$

### *name*

**Description** *Optional.* Specifies a symbolic name, as defined in an ACCESS, BIG, FILE, INTERFACE, SCREEN, SMALL, or TEXT statement. MANTIS clears the data referred to by the specified symbolic name.

**Format** Valid MANTIS symbolic name.

**Considerations**

- CLEAR with the symbolic name of a BIG or SMALL variable, or array, sets the value of the variable (or each element of the array) to zero.
- CLEAR with the symbolic name of a TEXT variable or array sets the current length of the variable (or each element of the array) to zero, giving it an empty value " ".
- CLEAR with the symbolic name of an entity defined by an ACCESS, FILE, INTERFACE, or SCREEN statement clears the values of all variables and arrays associated with the entity. In addition, MANTIS clears the value that is returned when the symbolic name is used to invoke a built-in text function.

### ALL

**Description** *Optional.* Clears the values of all variables and entities defined in the current program.

**Consideration** Data in programs or subroutines at higher or lower levels are not affected.

## TRAP

**Description** *Optional.* Clears a fault trap set by SET TRAP.

**Example**

```
10 BIG COUNT,SUBTOTAL(10)
20 FILE CUSTFILE("CUSTOMERS","XANADU")
30 SCREEN CUSTSCREEN("CUSTOMER")
40
50
60 CLEAR COUNT,SUBTOTAL,CUSTFILE,CUSTSCREEN
```

**General considerations**

- The CLEAR statement works differently in MANTIS for Windows and in MANTIS for the IBM mainframe. In compatibility mode, the CLEAR statement can be used only with a *screen-name* parameter or with no parameter.
- CLEAR with no parameter clears the logical display (except the heading in scroll mode). Program variables are not affected. The screen is cleared just before the next terminal I/O. See "MANTIS Logical Terminal Interface" on page 251 of this manual for additional information on how the CLEAR statement with no parameter works with the MANTIS Logical Terminal Interface.

## COMMIT

Use the COMMIT statement to indicate the completion of a Logical Unit of Work (LUW), or to enable or disable automatic COMMIT processing. Pending database updates are committed, preventing them from being backed out by RESET or a system failure.  Updated buffers for the MANTIS cluster and external PC files are flushed.

$$\textbf{COMMIT}\begin{bmatrix}\textbf{ON}\\ \textbf{OFF}\end{bmatrix}$$

### ON

**Description** *Optional.* Indicates that you want MANTIS to perform automatic COMMIT processing while awaiting terminal input (e.g., on every CONVERSE, OBTAIN, or WAIT statement).

**Consideration** The ON parameter is not supported by MANTIS for the IBM mainframe.

### OFF

**Description** *Optional.* Indicates that you do not want MANTIS to perform automatic COMMIT processing while awaiting terminal input.

**Consideration** The OFF parameter is not supported by MANTIS for the IBM mainframe.

**Example**

```
20 FILE CUST("CUSTOMER,"XANADU"),
30 'ORDERS("NEW_ORDERS","XANADU"),'
.
.
.
100  INSERT ORDERS
110  UPDATE CUST
120  COMMIT
```

## General considerations

- MANTIS may run slightly faster if COMMIT OFF is specified, but any system failure may lead to cluster corruption and lost data.
- COMMIT with no parameter flushes any updated buffers for the MANTIS cluster and for any external PC files that were opened by ACCESS statements. Any pending database updates are committed so they will not be backed out by a RESET or system failure.
- Automatic COMMIT processing is initially enabled.

## CONVERSE

Use the CONVERSE statement to display a map set consisting of one or more screens, add a screen map to the map set without displaying it, or remove a map from a map set. Screen fields show the values of corresponding MANTIS variables. Unprotected screen fields can be updated from the keyboard. Control returns to MANTIS when an action or macro key is pressed. When control returns to MANTIS, the MANTIS variables are updated.

See the discussion in “MANTIS Logical Terminal Interface” on page 251 for detailed examples of CONVERSE.

```
                         ┌                ┌        ┐  ┌ ┌         ┐ ┐   ┐
                         │                │ WAIT   │  │ │ WINDOW  │ │   │
                         │ [(row1,col1)]  │ SET    │  │ │ DISPLAY │ │   │
                         │                │ UPDATE │  └ └         ┘ ┘   │
                         │                └        ┘                    │
CONVERSE screen - name   │ [(row2,col2)] [template]                     │
                         │ RELEASE                                      │
                         └                                              ┘
```

### *screen-name*

**Description** *Required.* Specifies the symbolic name of a screen, as defined in a SCREEN statement.

**Format** MANTIS symbolic name as defined in a previously executed SCREEN statement.

## (*row1*,*col1*)

**Description** *Optional.* Specifies the position of the upper-left corner of the map in the logical display.

**Default** Upper-left corner of the logical display (1,1).

**Format** *row1* must be an arithmetic expression that evaluates to a row number in the logical display. Rows are numbered 1 to 32767, from top to bottom. *col1* must be an arithmetic expression that evaluates to a column number in the logical display. Columns are numbered 1 to 32767, from left to right.

**Considerations**

- The row and column values may be negative when you are specifying the position of a floating map in the logical display. This allows the floating map to be located relative to the bottom row (row 1 negative) or the right-hand edge (column 1 negative) of the physical screen.
- You can use negative values only if the corresponding windowing attribute of the map is disabled. To specify your screen design as a floating map, refer to the vertical and horizontal windowing map-level attributes in *MANTIS for Windows Facilities Reference Manual*, P19-2301.

## WAIT

**Description** *Optional.* Causes the specified map to become the active map in the map set. All other maps in the map set become passive maps. The contents of the window are not displayed on the screen.

**Consideration** CONVERSE *screen-name* WAIT adds the specified screen map to the map set, optionally specifying its (*row1*,*col1*) position. Other parameters are ignored.

## SET

**Description** *Optional.* Causes the specified map to become the active map in the map set. All other maps in the map set become passive maps. The contents of the window are displayed on the screen. Update of fields in passive maps is not allowed.

**Consideration** CONVERSE *screen-name* SET adds the specified map to the map set and causes the map set to be displayed. This is the active map in the map set (it will appear on top of the other (passive) maps).

## UPDATE

**Description** *Optional.* Causes the specified map to become the active map in the map set. All other maps in the map set become passive maps. The contents of the window are displayed on the screen. Update of fields in passive maps is allowed.

**Consideration** CONVERSE *screen-name* UPDATE acts the same as CONVERSE *screen-name* SET, except all data-entry fields that are fully displayed from underlying (passive) maps are unprotected. Unprotected fields that extend beyond the boundaries of the physical screen do not need to be completely displayed to be updated, provided the AUTOMATIC WINDOWING attribute is specified for the active map (refer to the *MANTIS for Windows Facilities Reference Manual*, P19-2301).

## WINDOW

**Description** *Optional.* Causes MANTIS to display the position of the window in the logical display. The row and column coordinates of the upper-left corner of the window are displayed in the position map at the bottom of the screen.

**Consideration** In MANTIS for the IBM mainframe, the WINDOW parameter selects window mode in which the program function keys are used for windowing functions instead of their normal purpose. In MANTIS for Windows, window mode is always in effect. Windowing functions can always be performed by using the logical windowing keys (see the table in “MANTIS editing and windowing keys” on page 32).

## DISPLAY

**Description** *Optional.* Has no effect in MANTIS for Windows, but allows (*row2,col2*) to be specified without specifying WINDOW.

## (*row2*,*col2*)

**Description** *Optional.* Specifies the position of the upper-left corner of the window in the logical display.

**Default** Upper-left corner of the logical display (1,1).

**Format** *row2* must be an arithmetic expression that evaluates to a row number in the logical display. Rows are numbered 1 to 32767, from top to bottom. *col2* must be an arithmetic expression that evaluates to a column number in the logical display. Columns are numbered 1 to 32767, from left to right.

## *template*

**Description** *Optional.* Specifies the name of a TEXT variable containing characters that are sent to the printer before the screen data.

**Considerations**

- The template parameter can be used to access printer capabilities that are not supported through the Screen Design Facility. You are responsible for assigning the correct escape sequences and associated data to the template text variable.
- The template parameter is not supported by MANTIS for the IBM mainframe.

## RELEASE

**Description** *Optional.* Removes the specified map from the map set. The resulting map set is not displayed until the next CONVERSE statement.

**Examples**

```
10 FILE RECORD("INDEX","SERENDIPITY")
20 SCREEN MAP("INDEX")
30 UNTIL RECORD="END" OR MAP="CANCEL"
40 .GET RECORD
50 .CONVERSE MAP
60 END
```

```
10 SCREEN CUSTOMER("CUSTOMER")
20 SCREEN SALESMAN("SALESMAN")
30 CONVERSE CUSTOMER_WAIT
40 CONVERSE SALESMAN(10,1) UPDATE
80 CONVERSE SALESMAN_RELEASE
```

## General considerations

- ♦ CONVERSE *screen-name* without a WAIT, SET, or UPDATE creates a new map set. This map set contains only the named screen and causes a physical I/O to occur, thereby displaying the map on your screen.

- ♦ To create a map set with more than one map, use one or more CONVERSE statements with the WAIT parameter, followed by one CONVERSE statement with either the SET or the UPDATE parameter.

- ♦ When you issue a CONVERSE statement with no parameter, MANTIS:

  - Determines which fields in each map are visible in the window, taking into account the front-to-back order of the map set. Some fields may be overlaid by fields in maps that are in front of them.

  - Sets the contents of each data field to the value of the corresponding variable. Numeric values are formatted using the edit mask for the field if it has one; otherwise, the standard decimal format is used.

  - Determines which data fields will be protected to prevent them from being updated. A field is protected if it is partially overlaid by another field or if it is partially outside the window, unless it is in a map that has the AUTOMATIC WINDOWING attribute. All fields in a passive map are protected unless the UPDATE parameter is used.

- ♦ When conversing maps using the UPDATE option of the CONVERSE statement, the following rule governs whether you can update partially displayed fields on a passive map:

  - Fields that are partially displayed because they are overlaid by the active map cannot be updated.

  - Fields on a passive map that are partially displayed because they overlap the physical screen boundary can be updated.

Fields on an active map that are partially displayed because they overlap the physical screen cannot be updated unless the AUTOMATIC WINDOWING attribute is specified for the active map. In MANTIS for the IBM mainframe, they can always be updated with a CONVERSE UPDATE.

- Sets up the Message field on the next to last line of the screen using any data that is pending from the most recent SHOW statement.
- Displays the contents of the window on the screen and waits for your response.

♦ Since MANTIS reserves the last two lines on all screens (unless you specify the Full Display attribute in Screen Design or with an ATTRIBUTE statement), they will appear to be blank. The last line has two fields: the Unsolicited Input field and Key Simulation field. You can enter data on the last line by positioning the cursor:

  - In the first field to enter data (see “OBTAIN” on page 170).
  - In the second field to simulate a program function key (e.g., if you enter “PF7”, MANTIS assumes you pressed PF7). You can also use the second field on the last line to terminate a program (e.g., enter KILL).

♦ See the discussion in “MANTIS Logical Terminal Interface” on page 251 for detailed examples of CONVERSE. Before returning control to the program, MANTIS:

  - Uses the contents of each updated data field to update the value of the corresponding MANTIS variable. Numeric fields are converted using the edit mask for the field if it has one; otherwise, the standard decimal conversion is performed. MANTIS performs any extended edit validation (such as range checking) that you specified in Screen Design. If numeric conversion or validation fails, MANTIS displays an error message on the next to last line and highlights the fields to be corrected.
  - Saves a text value identifying the action key that you entered (or the value that you entered in the Key Simulation field). This value can be obtained by using the symbolic name of the active map to invoke a built-in text function.

- A map added to a map set in which the map already exists changes the ordering within the map set.  If you specify SET or UPDATE, the added map will move from passive to active.
- You can display a map only once within a map set.  If you add a map twice (with a WAIT, SET, or UPDATE option), it moves to the new logical display position.  If you want the same screen image to appear twice, you must have two screen variables and converse both of them.  For example,

  ```
  SCREEN MAP1("INDEX"),MAP2("INDEX",PREFIX)
  CONVERSE MAP1 WAIT
  CONVERSE MAP2(20,30)SET
  ```

  Unless you prefix one of the maps, the variables will be automatically mapped between the two maps.  If you do not prefix one of the maps, you may have two screen fields assigned to one variable.  The final value of the variable (following the CONVERSE) will be the value last processed, because screen fields are processed left to right and top to bottom.
- If a map set (or part of a map set) is conversed directly via external DO, MANTIS will clear the entire map set when it exits from the external DO.  To avoid clearing the portion of the map set conversed in the calling program, release the external map using the CONVERSE map RELEASE statement.

## COS

Use the COS function to return the cosine of an arithmetic expression in radians.

**COS(*a*)**

***a***

**Description** *Required.* Specifies the number whose cosine you want returned.

**Format** Valid arithmetic expression.

**Example**

```
COS(10)   returns -.839071529
```

## CURSOR

Use the CURSOR function to determine whether the cursor was in a specified field when the last action key was pressed.  MANTIS returns TRUE if the cursor appeared within the domain of the field you specified when an action key was pressed.  Otherwise, MANTIS returns FALSE.

**CURSOR(*screen-name*,*field-name*)**

### *screen-name*

**Description** *Required.*  Specifies the symbolic name of the screen you are testing, as defined in a SCREEN statement.

**Format** MANTIS symbolic name.

**Consideration** CURSOR does not check to see whether the screen was actually part of the map set on the last CONVERSE operation.

### *field-name*

**Description** *Required.*  Specifies the symbolic name of the field you are testing, as defined in Screen Design.

**Format** MANTIS symbolic name.

**Examples**

- Allow user to select items from a list by cursor positioning.  For example, to allow selection of an item from a list of choices (for menu processing):

```
WHEN CURSOR(MENU_SCREEN,CLIENT_UPDATE)
.DO CLIENT_PROCESSING
WHEN CURSOR(MENU_SCREEN,INVENTORY_UPDATE)
.DO INVENTORY_PROCESSING
.
.
.
END
```

- Allow the cursor position to control scrolling of a group of lines. For example, the following code redisplays a list of records from the record selected by the cursor position:

```
WHILE I<=COUNT
.IF CURSOR(CLIENT_SCREEN,NAME(I))
..FIRST_KEY=NAME(I)
..I=COUNT:| END LOOP
.END
.I=I+1
.END
GET REC(FIRST_KEY)LEVEL=1
I=1
WHILE I<COUNT AND REC<>"END"
.I=I+1
.GET REC LEVEL=I
END
```

## DATAFREE

Use the DATAFREE function to return the number of bytes currently available in your data area. This function is provided for compatibility with MANTIS for the IBM mainframe. Its value is meaningless on MANTIS for Windows.

**DATAFREE**

### Example

```
SHOW DATAFREE
```

### General considerations

- ♦ The DATAFREE function is present for compatibility with MANTIS for the IBM mainframe.
- ♦ The value returned represents the number of bytes available in the data areas that are currently allocated. More data areas will be allocated if required.

# DATE (function)

Use the DATE function to return the text value of the current date. The format of the date is determined by a previous date mask specification using the DATE statement.

**DATE**

## Example

```
30 ...TEXT LAST_UPDATED(20)
40 ...DATE="YYYY/MM/DD"
50 ...WHEN MAP="ENTER"
60 ....ERROR=FALSE
70 ....DO EDIT_FIELDS
80 ....IF ERROR=FALSE
90 .....LAST_UPDATED=DATE
100 .....UPDATE REC
110 ....END
.
.
.
```

## General considerations

- The current date mask setting is inherited on an external DO or CHAIN LEVEL, and is restored upon subprogram EXIT. This inheritance does not occur when a nonprivileged user's program attempts to externally DO or CHAIN LEVEL a privileged user's program.
- A CHAIN (without LEVEL) will reset the date mask to NULL (“”).
- For nonprivileged user's programs, if the date mask has not been previously specified, or has been set to NULL (“”), the DATE function returns the current date in the format specified by the DATE MASK general option in the configuration file.
- For privileged user's programs, if the date mask has not been previously specified, or has been set to NULL (“”), the DATE function returns the current time in *YY*/*MM*/*DD* format.
- See DATE (statement), TIME (function), and TIME (statement).

## DATE (statement)

Use the DATE statement to specify the date mask to be used by the DATE function.

**DATE=*mask-expression***

### *mask-expression*

**Description** *Required.* Specifies the date mask to be used by the DATE function.

**Format** Text expression of 0–32 characters.

**Considerations**

- ♦ If the mask specified is longer than 32 characters, the first 32 characters are used.
- ♦ The following special strings are substituted with the data described when the DATE function is invoked. Any character in the mask that is not part of one of these special strings is treated as punctuation and is not translated.
  - *DDD* Day of the year (Julian day 001-366)
  - *DD* Day of the month (01-31)
  - *MMM* Name of the month (JAN-DEC)
  - *MM* Month (01-12)
  - *YYYY* 4-digit year (0000–9999)
  - *YY* 2-digit year (00–99)
- ♦ Where there are ambiguities on substitution, the substitution takes place in the order of the special strings listed above.
- ♦ There is a special mask escape character (%). Any character following this escape is taken literally and the escape can be used to break up what might otherwise be a valid substitution string.
- ♦ When alphabetic substitution is required (e.g., MMM), the case of any special string character is reproduced on substitution. Case is not important for numeric substitution.

## Example

```
DATE="DD-Mmm-YYYY"
SHOW DATE
18-Aug-1997

DATE="Summer of ‘YY"
SHOW DATE
Su08er of ‘97

DATE="Sum%mer of ‘YY"
SHOW DATE
Summer of ‘97
```

## General considerations

- See DATE (function), TIME (function), and TIME (statement).

## DELETE

Use the DELETE statement to delete a record from a MANTIS file or from an external file. Before you delete a record, you must first define it with a FILE or ACCESS statement. You need not read a record from a file before deleting it.

### MANTIS file delete

**DELETE *file-name* [LEVEL=*level-number*]**

#### *file-name*

| | |
|---|---|
| **Description** | *Required.* Specifies the symbolic name of a MANTIS file from which you want to delete a record. |
| **Format** | MANTIS symbolic name defined in a previously executed FILE statement. |

#### LEVEL=*level-number*

| | |
|---|---|
| **Description** | *Optional.* Specifies which level of MANTIS array element(s) contains the key field(s) of the record you want to delete. |
| **Default** | LEVEL=1 |
| **Format** | Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding FILE statement. |

## Example

```
10 ENTRY INDEX
20 .FILE RECORD("INDEX","SERENDIPITY")
30 .SCREEN MAP("INDEX")
40 .GET RECORD
50 .WHILE RECORD<>"END" AND MAP<>"CANCEL"
60 ..CONVERSE MAP
70 ..WHEN MAP="PF1"
80 ...INSERT RECORD
90 ..WHEN MAP="PF2"
100 ...DELETE RECORD
110 ..WHEN MAP="PF3"
120 ...UPDATE RECORD
130 ..END
140 ..GET RECORD
150 .END
160 EXIT
```

## General considerations

- Before you delete a record from a MANTIS file, you must first open it with a FILE statement. You do not need to read the record before deleting it.

- The contents of key data elements identify the record to be deleted.

- The status of the DELETE can be obtained by using the symbolic name *file-name* to invoke a built-in text function. The status is a null text value if the DELETE was successful, or held if MANTIS could not update the record because another user has locked the record with the GET ENQUEUE statement. If the DELETE is unsuccessful, you can use the HELP LAST command (in Program Design) to examine the error message that may have been returned. This information is also available to MANTIS fault handling routines (see “SET” on page 202). MANTIS may also return the following status if TRAP is in effect:

| | |
|---|---|
| LOCK | The password specified in the FILE statement is invalid or the record is locked by another user. |
| ERROR | A physical error occurred while retrieving the record. |

- MANTIS performs an implicit COMMIT on every terminal input operation, unless this functionality is disabled by the COMMIT OFF statement.

## External file delete

**DELETE *access-name* [LEVEL=*level-number*]**

### ***access-name***

**Description** *Required.* Specifies the symbolic name of an external file from which you want to delete a record.

**Format** MANTIS symbolic name defined in a previously executed ACCESS statement.

**Consideration** If the file has been released, it will be automatically reopened, but the position in the file will be lost. For example, if a GET...NEXT is executed subsequently, the first record in the file is returned regardless of which record was being accessed before the release.

### **LEVEL=*level-number***

**Description** *Optional.* Specifies which level of MANTIS array element(s) contains the key field(s) or the reference variable of the record you want to delete.

**Default** LEVEL=1

**Format** Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding ACCESS statement.

**Example**

```
20 .ACCESS RECORD("INDEX","SERENDIPITY",16)
30 .SCREEN MAP("INDEX")
40 .CONVERSE MAP
50 .COUNTER=1
60 .WHILE MAP<>"CANCEL" AND COUNTER <17
70 ..WHEN INDICATOR(COUNTER)="G"
80 ...GET RECORD LEVEL=COUNTER
90 ..WHEN INDICATOR(COUNTER)="D"
100 ...DELETE RECORD LEVEL=COUNTER
.
.
.
.
```

## General considerations

- Before you delete a record from an external file, you must first define it with an ACCESS statement (see “ACCESS” on page 75). You do not need to read the record before deleting it.
- For INDEXED files, the contents of key data elements identify the record to be deleted.
- You cannot delete records from SEQUENTIAL files.
- You cannot delete records from DOS NUMBERED files.
- For VSAM NUMBERED files, the Relative Record Number (RRN) contained in the corresponding reference variable identifies the record to be deleted.
- The status of the DELETE can be obtained by using the symbolic name *access-name* to invoke a built-in text function. The status is a null text value if the DELETE was successful, or held if MANTIS could not update the record because another user has locked the record with the GET ENQUEUE statement. If the DELETE is unsuccessful, you can use the HELP LAST command (in Program Design) to examine the error message that may have been returned. This information is also available to MANTIS fault handling routines (see “SET” on page 202). MANTIS may also return a status of LOCK or ERROR if TRAP is in effect.

| | |
|---|---|
| LOCK | The password specified in the ACCESS statement for this file view is not valid for deletions or insertions. |
| ERROR | A physical error occurred while deleting the record or the RRN for NUMBERED files specified a record number that does not exist, or an error occurred while opening the file. |

- MANTIS performs an implicit COMMIT on every terminal input operation, unless this functionality is disabled by the COMMIT OFF statement.

## DEQUEUE

The DEQUEUE statement is provided for compatibility with MANTIS for the IBM mainframe.  It has no effect on MANTIS for Windows.

**DEQUEUE *resource***

### *resource*

**Description** *Required.* Specifies the resource you want to release.

**Format** Text expression.

**Example**

```
20 FILE REC("CUSTOMERS","ALIBABA")
.
.
.
100 ENQUEUE "CUSTOMERS"+RECORD_KEY
110 GET REC(RECORD_KEY)
.
.
.
200 UPDATE REC
210 DEQUEUE "CUSTOMERS"+RECORD_KEY
```

## DO

Use the DO statement to execute an internal or external subroutine. A subroutine is a block of statements enclosed by ENTRY-EXIT statements. An internal subroutine is part of the current program. An external subroutine is in a separate program that must be identified by a PROGRAM statement. When MANTIS encounters the EXIT statement in the subroutine, it returns to execute the statements following the DO statement:

$$\textbf{DO}\ \left\{\begin{array}{l}\textbf{\textit{entry - name}}\\ \textbf{\textit{program - name}}\end{array}\right\}\left[(\textbf{\textit{argument}},\ldots)\right]$$

### *entry-name*

| | |
|---|---|
| **Description** | *Optional.* Specifies the name of a subroutine, as defined in an ENTRY statement (for an internal subroutine). |
| **Format** | MANTIS symbolic name. |

### *program-name*

| | |
|---|---|
| **Description** | *Optional.* Specifies the name of a subroutine, as defined in a PROGRAM statement (for an external subroutine). |
| **Format** | MANTIS symbolic name. |

### *argument*

| | |
|---|---|
| **Description** | *Optional.* Specifies an argument you want passed to the subroutine. |
| **Format** | MANTIS symbolic name, expression, or array element. |
| **Consideration** | You can pass 1–255 arguments. |

## Example

```
100 ENTRY EDIT PROGRAM
.
150 .PROGRAM EDIT RTN("VALIDATION","COMMON")
160 .TYPE="CREDIT CHECK"
170 .DO EDIT RTN(TYPE,CUST_NO,STATUS,MESSAGE)
180 .IF STATUS<>"GOOD"
190 ..DO ERROR RTN(CUST_NO)
200 .END
210 .TYPE="SELECT SALES REP"
220 .DO EDIT RTN(TYPE,CUST_NO,STATUS,MESSAGE)
230 .IF STATUS<>"GOOD"
240 ..DO ERROR RTN(SALES_REP)
250 .ELSE
260 ..SALES_REP=MESSAGE
270 .END
280 EXIT
290 ENTRY ERROR_RTN(FIELD)
300 .IF NOTE=""
310 ..NOTE=MESSAGE
320 ..ATTRIBUTE(MAP,FIELD)="BRI,CUR"
330 .ELSE
340 ..ATTRIBUTE(MAP,FIELD)="BRI"
350 .END
360 EXIT
```

## General considerations

- To maintain compatibility with MANTIS for the IBM mainframe, do not use expressions as arguments.
- You must include ENTRY-EXIT statements around a subroutine, but the *entry-name* on the ENTRY statement is used only to DO an internal subroutine.  The *program-name* on the PROGRAM statement is used to DO an external subroutine.
- The subroutine ENTRY statement cannot have parameters defined as arrays.

- If the main part of your program does not have ENTRY-EXIT statements, a STOP statement is required as the last executed statement in the main program.
- Each argument is either the symbolic name of a MANTIS variable or an expression:
  - If the argument is a symbolic name, each reference to the corresponding argument in the subroutine uses the specified variable ("call by reference"). Values assigned to the argument in the subroutine are stored in the specified variable.
  - If the argument is an expression, it is evaluated before calling the subroutine and each reference to the corresponding parameter in the subroutine uses the value of the expression ("call by value"). Values assigned to the argument in the subroutine are lost on return from the subroutine. A subscripted symbolic name is an expression in this context.
- The argument(s) in the DO statement must correspond in number with the parameters on the ENTRY statement.
- Since all variables previously defined by the calling routine are available to an internal subroutine, arguments are used only to set up alias names.
- Only variables explicitly passed as arguments are available to an external subroutine. A symbolic name defined in a SCREEN, FILE, ACCESS, or INTERFACE statement may be passed (for use in CALL, CONVERSE, GET, UPDATE, INSERT, or DELETE statements), but this will not give access to the implicitly defined variables in this first group of statements.
- If you stop execution of a subroutine, enter SHOW DOLEVEL to see where you are. Use the UP command to return to the calling routine.
- See "External DO" on page 267 for additional information on external DO.

## DOLEVEL

Use the DOLEVEL function to return the current level in an external subroutine.

**DOLEVEL**

### Example

```
SHOW DOLEVEL
```

### General considerations

- The original program has a DOLEVEL of zero.  With each nested external DO statement, the DOLEVEL value increases by one.
- Use the SHOW DOLEVEL command when you are debugging to see which level you are executing.  You can modify and replace programs at any level.  The revised version is executed in the next RUN.
- The DOWN and UP commands enable you to list and edit an external subroutine in the work area.  DOWN selects an external subroutine at a lower level.  UP selects an external subroutine at a higher level.  (Refer to the *MANTIS for Windows Facilities Reference Manual*, P19-2301, for more information.)

## E

Use the E function to return the value of natural logarithm base e (2.71828182846).

**E**

### Example

```
10 X=2*E+1
```

## EDIT

Use the EDIT statement to edit a text variable or array.  The current value of the text variable or array is written to a temporary PC file, and your editor is then invoked to edit the temporary file.

**EDIT LIST *text-name***

### LIST

**Description** *Required.*  Specifies that you want to edit a text variable or array.

### *text-name*

**Description** *Required.*  Specifies the symbolic name of the text variable or array that you want to edit.

**Consideration** An array can have no more than one dimension.

**Example**

```
EDIT LIST SURNAMES
```

**General considerations**

- ♦ The EDIT statement is not supported by MANTIS for the IBM mainframe.
- ♦ For a variable, the text value occupies one line in the file.  For an array, the text values of successive array elements occupy successive lines in the file.
- ♦ When MANTIS reloads the temporary file into the variable or array, tabs are expanded using the value specified in the EDITTABS environment variable, lines that exceed the maximum length of the text variable or array element are truncated, excess lines are discarded, and array elements for which there are not enough lines are cleared.
- ♦ The command line for the external editor is specified in the EDITOR environment variable or in the configuration file.
- ♦ The name and path of the temporary operating system file is specified in the EDITFILE and TMP environment variables.
- ♦ For a full description of the environment variables used by MANTIS, refer to the *MANTIS for Windows Administration Guide*, P19-2304.

## ENQUEUE

The ENQUEUE statement is provided for compatibility with MANTIS for the IBM mainframe. It has no effect in MANTIS for Windows.

**ENQUEUE *resource***

### *resource*

**Description** *Required.* Specifies the resource you want to hold.

**Format** Text expression.

**Example**

```
20 FILE REC("CUSTOMERS","ALIBABA")
.
.
.
100 ENQUEUE "CUSTOMERS"+RECORD_KEY
110 GET REC(RECORD_KEY)
.
.
.
200 UPDATE REC
210 DEQUEUE "CUSTOMERS"+RECORD_KEY
```

## ENTRY-EXIT

Use ENTRY-EXIT statements to define the beginning and end of a subroutine or program. When a DO or CHAIN statement calls a subroutine or program, the arguments passed by the DO or CHAIN statement replace the corresponding parameters specified on the ENTRY statement.

**ENTRY *entry-name*[(*parameter*,...)]**

**.**

**statements**

**.**

**EXIT**

### *entry-name*

| | |
|---|---|
| **Description** | *Required.* Specifies a name to identify the subroutine or program. |
| **Format** | MANTIS symbolic name. |

### *parameter*

| | |
|---|---|
| **Description** | *Optional.* Specifies the names by which you will refer to any parameters passed to the subroutine or program. |
| **Format** | MANTIS symbolic name. |
| **Consideration** | You can specify 1–255 parameters. |

## Example

```
10 ENTRY DATA_ENTRY
20 .SCREEN MAP1("INDEX")
30 .FILE REC1("INDEX","SERENDIPITY")
40 .CONVERSE MAP1
50 .WHILE MAP1<>"CANCEL"
60 ..DO INSERT_RECORD
70 ..CLEAR MAP1
80 ..CONVERSE MAP1
90 .END
100 EXIT
110 ENTRY INSERT_RECORD
120 .INSERT REC1
130 EXIT
```

## General considerations

- When you pass data from one program to another, use ENTRY-EXIT statements for each program.
- The *entry-name* on the ENTRY statement is used to DO an internal subroutine. The *program-name* on the PROGRAM statement is issued to DO an external subroutine.
- Arguments on DO and CHAIN statements must correspond in number with the parameters specified on the ENTRY statement.
- EXIT returns control from a subroutine to the calling program, and is equivalent to the STOP statement if executed in a mainline program.
- ENTRY-EXIT statements cannot be nested, but each program can have multiple ENTRY-EXIT procedures, the first set being the top-level routine.
- You cannot use the ampersand (&) for indirect reference on either the *entry-name* or parameters.
- Internal subroutines have access to all the variables defined in the surrounding program context. Be careful not to confuse ENTRY statement arguments with global program variables, since the global variables having the same name as ENTRY arguments will not be accessible within the scope of the subroutine.
- See "MANTIS Logical Terminal Interface" on page 251 for more details on this statement.

## EXP

Use the EXP (exponent) function to return the value of e to the power of an arithmetic expression (inverse of the LOG function).

**EXP(*a*)**

***a***

**Description** *Required.* Specifies the number whose exponential function you want returned.

**Format** Valid arithmetic expression.

**Example**

`EXP(1)` returns `2.71828182846`

# FALSE

Use the FALSE function to return the value FALSE (zero).

**FALSE**

## Example

```
10 ENTRY CUST_ENTRY
20 .SCREEN MAP("CUST_ENTRY")
30 .FILE REC("CUST_FILE",PASSWORD)
40 .FILE ST_CODES("STATE_CODES",PASSWORD)
50 .CONVERSE MAP
60 .WHILE MAP<>"CANCEL"
70 ..ERROR=FALSE
80 ..DO VALIDATE_INFO
90 ..IF ERROR=FALSE
100 ...INSERT REC
110 ...CLEAR MAP
120 ..END
.
.
.
```

## FILE

Use the FILE statement to define a MANTIS file that your program will access.  MANTIS retrieves the file profile from the library and defines a MANTIS variable or array with the same name as each field in the file profile.

**FILE *file-name*(*libname*,*password*[,PREFIX][,*levels*]),...**

### *file-name*

**Description**

*Required.*  Defines the symbolic name by which you will refer to the file in subsequent GET, UPDATE, INSERT, and DELETE statements.

**Format**

Unique MANTIS symbolic name.

**Considerations**

- No processing occurs if *file-name* is already defined.
- The *file-name* symbolic name returns a text value indicating the status of the most recent GET, UPDATE, INSERT, or DELETE operation performed on a MANTIS file.

**Example**

```
200 ENTRY VALIDATE_INFO
210 .MESSAGE=""
220 .FILE SHADOW_FILE("CUST_FILE",PASSWORD,PREFIX)
230 .GET SHADOW_FILE(CUST-NUMBER)
240 .IF SHADOW_FILE="FOUND"
250 ..ATTRIBUTE(MAP,CUST_NUMBER)="BRI,CUR"
260 ..MESSAGE="DUPLICATE CUSTOMER NUMBER"
270 .END
.
.
.
```

## ***libname***

| | |
|---|---|
| **Description** | *Required.* Specifies the name of the file profile as it was saved during File Design. |
| **Format** | Text expression that evaluates to [*user name*:]*file-name*. |
| **Consideration** | If the file profile is in your own library, you do not need to specify user name. |

## ***password***

| | |
|---|---|
| **Description** | *Required.* Specifies the password for the type of file access (GET, UPDATE, INSERT/DELETE) your program requires. |
| **Format** | Text expression that evaluates to the password specified during File Design. |

## **PREFIX**

**Description** *Optional.* Specifies the prefix *file-name*_ for all variable names associated with this file.

For example, if the MANTIS file BOTTLES has a field named VOLUME, the following statement causes the corresponding variable BIN_VOLUME to be defined in the work area:

```
10 FILE BIN("BOTTLES","MAGIC",PREFIX)
```

## ***levels***

**Description** *Optional.* Specifies the number of levels you want to use in GET, UPDATE, INSERT, and DELETE statements for this file.

**Default** 1

**Format** Arithmetic expression that evaluates to a value in the range 1–255.

**Considerations**

- If you do not specify *levels*, MANTIS defines a BIG, SMALL, or TEXT variable or array for each field in the file profile according to the type and dimensions specified during File Design.
- If you specify *levels*, MANTIS adds an extra dimension to each field to allow a separate value to be stored at each level. MANTIS defines a one-dimensional array instead of a variable, a two-dimensional array instead of a one-dimensional array, and so on. The first dimension of each array is *levels*.
- If you specify *levels*, you must also specify LEVEL=*level-number* (unless you want the default LEVEL=1) in all GET, UPDATE, INSERT, and DELETE statements for the file.

## Examples

♦ 20 .FILE RECORD("INDEX","SERENDIPITY",16)

```
30 .SCREEN MAP("INDEX")
40 .WHILE RECORD<>"END" AND MAP<>"CANCEL"
50 ..CLEAR MAP:LEVEL_NUMBER=1
70 ..GET RECORD LEVEL=LEVEL_NUMBER
80 ..WHILE RECORD<>"END" AND LEVEL_NUMBER<16
90 ...LEVEL_NUMBER=LEVEL_NUMBER+1
100 ...GET RECORD LEVEL=LEVEL_NUMBER
110 ..END
120 ..CONVERSE MAP
130 .END
```

♦ 20 FILE RECORD("EXAMPLE:INDEX","SERENDIPITY",PREFIX,16)

♦ 20 TEXT PASS(16)

```
30 TEXT FILENAME(33)
40 PASS="SERENDIPITY"
50 FILENAME="EXAMPLE:INDEX"
60 FILE RECORD(FILENAME,PASS,PREFIX,16)
```

The last two examples produce the same result.

## FOR-END

Use the FOR-END statements to execute a block of statements repeatedly while a counter is incremented or decremented through a specified range of values.

**FOR *counter* = *initial* TO *final* [BY *increment*]**

**.**

**statements**

**.**

**END**

### *counter*

| | |
|---|---|
| **Description** | *Required.* Specifies the numeric variable or array element that MANTIS will use as a counter. |
| **Format** | Name of a BIG or SMALL variable or the subscripted name of a BIG or SMALL array. |

### *initial*

| | |
|---|---|
| **Description** | *Required.* Specifies the initial value of the counter. |
| **Format** | Arithmetic expression. |

### *final*

| | |
|---|---|
| **Description** | *Required.* Specifies the final value with which the counter is compared. |
| **Format** | Arithmetic expression. |

### BY *increment*

| | |
|---|---|
| **Description** | *Optional.* Specifies the increment that is added to the counter after each execution of the block of statements. |
| **Default** | BY 1 |
| **Format** | Arithmetic expression. |

## Example

```
10 FOR I=1 TO SIZE(ARRAY,1)
20 .SHOW ARRAY(I)
30 END

50 FOR J=SIZE(STRING) TO 1 BY -1
60 .IF STRING(J,J)<>" "
70 ..BREAK
80 .END
90 END
```

## General considerations

- MANTIS sets the counter to the initial value that is evaluated only once. MANTIS evaluates the final value and the increment prior to each execution of the block of statements.
- MANTIS repeatedly executes the block of statements while the counter is less than or equal to the final value and the increment is positive, or the counter is greater than or equal to the final value and the increment is negative. After executing each block of statements, MANTIS adds the increment to the counter.
- If the block of statements changes the values of the counter, the final value, or the increment, this will be reflected in the next comparison of the counter and the final value.
- The logic of FOR-END statements can be expressed in terms of equivalent WHILE-END statements, as follows:

```
10 FOR COUNTER=INITIAL TO FINAL BY INCREMENT
20 .SHOW COUNTER
30 END

40 COUNTER=INITIAL
50 WHILE INCREMENT>0 AND COUNTER<=FINAL
60 ' OR INCREMENT<0 AND COUNTER>=FINAL
70 .SHOW COUNTER
80 .COUNTER=COUNTER+INCREMENT
90 END
```

## FORMAT

Use the FORMAT function to return the text value resulting from the conversion of the specified number using the specified edit mask.  (Refer to the *MANTIS for Windows Facilities Reference Manual*, P19-2301.) You can also use the FORMAT function to provide formatted output for SHOW fields or fields on a screen or file not normally formatted.

**FORMAT(*number*, *mask*)**

### *number*

**Description** *Required.*  Specifies the number you want converted using the specified edit mask.

**Format** Valid arithmetic expression.

### *mask*

**Description** *Required.*  Specifies the edit mask.

**Format** Text expression that evaluates to a valid edit mask.

**Considerations**

- For compatibility with MANTIS for the IBM mainframe, you can use Z instead of 0, but Z is treated as a fill or float character if repeated in consecutive positions.  Note that Z must be in uppercase.
- When using edit masks, CR, DB, and DR must be entered in uppercase.

**Examples**

- To test your screen design masks for expected results:

  ```
  FORMAT (46.35,"$******.##") returns $****46.35
  ```

- To provide formatted output for SHOW fields or fields on a screen or file that are not normally formatted:

  ```
  SHOW NAME,AT(40),FORMAT(ACCT NO,"Z##-####-#"), AT(55),
  FORMAT(BALANCE,"##,##Z.##CR")
  ```

  produces this result:

  ```
  HENRIETTA JOHNSON                  034-4783-1     127.89CR
  ```

## FSI

The FSI function indicates the success or failure of an external file access or of an interface CALL.

**FSI(*name*[,*msg*])**

### *name*

|  |  |
|---|---|
| **Description** | *Required.* Specifies the symbolic name of an external file or interface. |
| **Format** | MANTIS symbolic name. |

### *msg*

|  |  |
|---|---|
| **Description** | *Optional* for external files; not allowed for interfaces. Specifies the symbolic name of the variable to receive the operating system error message. |
| **Format** | MANTIS symbolic name specified as a text variable with a length of at least 40. |

## Examples

-
  ```
  ACCESS X("TEXT","PSW")
  20 TEXT MESSAGE(40)
  30 TRAP X ON
  40 GET X FIRST
  50 IF X="ERROR"
  60 .IF FSI(X,MESSAGE)="UNAVAILABLE"
  70 ..SHOW "FILE TEST IS UNAVAILABLE"
  80 .ELSE
  90 ..SHOW "FILE TEST GET FAILED:STATUS="+MESSAGE
  100 .END
  120 END
  ```

-
  ```
  20 INTERFACE ITF("INTERFACE1","ALIBABA")
  30 TRAP ITF ON
  40 CALL ITF
  50 IF ITF<>""
  60.SHOW ITF,FSI(ITF)
  70 END
  ```

## General consideration

Function Status Indicators (FSIs) reflect the success or failure of your command.  See “CALL” on page 90 or refer to the *MANTIS for Windows Facilities Reference Manual*, P19-2301, for more details on FSIs and interfaces.

# GET

Use the GET statement to read a record from a MANTIS file or an external file. MANTIS copies the fields in the record to the corresponding MANTIS variables and arrays. Before you can read a record from a MANTIS file, you must define the file with a FILE statement.

## MANTIS file GET

```
                ┌                  ┌       ┐ ┐
                │ (key,...)        │ NEXT  │ │
                │                  │ EQUAL │ │
                │                  └       ┘ │
GET file-name   │ FIRST                      │ [ENQUEUE][LEVEL = level-number]
                │ NEXT                       │
                └                            ┘
```

### *file-name*

**Description** *Required.* Specifies the symbolic name of the file you want to access, as defined in a FILE statement.

**Format** MANTIS symbolic name.

### (*key*,...)

**Description** *Optional.* Specifies the key of the desired record. MANTIS assigns the value of each key to the corresponding key element in the file profile (the first key parameter to the first key element, the second key parameter to the second key element, etc.).

**Format** Expression of the same type and size as the corresponding key element in the file profile.

**Consideration** You may specify fewer key parameters than there are key elements in the file profile. MANTIS then retrieves the first record in which the corresponding key elements are equal to or greater than the partial key. When you use a partial key, MANTIS returns a status of NEXT unless it reaches the end of file (when it returns a status of END).

## FIRST

**Description** *Optional.* Retrieves the first record in the file.

## NEXT

**Description** *Optional.* Retrieves the next record in the file.

**Consideration** NEXT is the default option for the GET statement.

## (*key*,...) NEXT

**Description** *Optional.* Retrieves the next record in the file with a key greater than the specified key.

## (*key*,...) EQUAL

**Description** *Optional.* Retrieves a record only if the specified key parameters equal the corresponding key elements. MANTIS returns FOUND if the record exists; otherwise, it returns NOTFOUND.

**Considerations**

- When EQUAL is not specified, MANTIS will retrieve the next record in key sequence and return NEXT if it cannot find an equal key.
- When GET...EQUAL is issued, a subsequent GET...NEXT returns the next record (in key sequence), even if a NOTFOUND status was returned on the GET...EQUAL.

## ENQUEUE

**Description** *Optional.* Obtains a lock on the record. The lock will be released at the next commit point (e.g., at an explicit COMMIT statement, at an implicit commit during screen I/O, when the file is closed with a RELEASE statement, or when the program terminates with a CHAIN or returns to the main facility menu). The lock will also be released if the record is updated or deleted.

**Consideration** A lock is obtained only if MANTIS is running in shared mode. Refer to the *MANTIS for Windows Administration Guide*, P19-2304, for a description of the /SHARE command line parameter.

## LEVEL=*level-number*

**Description** *Optional.* Specifies which level of MANTIS array elements the fields in the record are copied to.

**Default** LEVEL=1

**Format** Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding FILE statement.

**Example**

```
20 .FILE RECORD("INDEX","SERENDIPITY",16)
30 .SCREEN MAP("INDEX")
40 .WHILE RECORD<>"END" AND MAP<>"CANCEL"
50 ..CLEAR MAP:LEVEL NUMBER=1
70 ..GET RECORD("WILLIAMS") LEVEL=LEVEL_NUMBER
80 ..WHILE RECORD<>"END" AND LEVEL_NUMBER<16
90 ...LEVEL_NUMBER=LEVEL_NUMBER+1
100 ...GET RECORD LEVEL=LEVEL_NUMBER
110 ..END
120 ..CONVERSE MAP
130 .END
```

### General considerations

- Before you can read a record from a MANTIS file, you must open it with a FILE statement.
- In MANTIS for the IBM mainframe, the ENQUEUE parameter of the GET statement is not allowed for MANTIS files and the NEXT parameter is not allowed when a key is also specified for MANTIS files.

- The status of the GET can be obtained by using the symbolic name *file-name* to invoke a built-in text function.  The status can be one of the following values:

| | |
|---|---|
| FOUND | MANTIS successfully retrieved the record identified by the key you supplied. |
| NEXT | MANTIS retrieved the next record in sequence for a GET without a key, or MANTIS retrieved the next record in key sequence when it could not locate the supplied key, or when a partial key was supplied. |
| END | MANTIS failed to retrieve the record because it reached the end of the file. |
| NOTFOUND | EQUAL was specified and MANTIS could not find a record with an equal key. |
| HELD | MANTIS failed to retrieve the record because it was held by another user. |

The following values may also be returned if TRAP is in effect:

| | |
|---|---|
| LOCK | The password specified in the FILE statement is invalid or the record is locked by another user. |
| ERROR | A physical error occurred while retrieving the record. |
| DATA | The record retrieved contained invalid data. |

- If the GET is unsuccessful, you can use the HELP LAST command (in Program Design) to examine the system error message that may have been returned.  This information is also available to MANTIS fault handling routines (see “SET” on page 202).

## External file GET

```
                    ┌                    ┐
                    │ (key,...) ┌NEXT ┐  │
                    │           └EQUAL┘  │
GET access - name   │ FIRST              │ [ENQUEUE][LEVEL =  level - number]
                    │ NEXT               │
                    └                    ┘
```

***access-name***

**Description** *Required.* Specifies the symbolic name of the file you want to access, as defined in an ACCESS statement.

**Format** MANTIS symbolic name.

***key*,...**

**Description** *Optional.* Specifies the key of the desired record. MANTIS assigns the value of each key to the corresponding key element in the external file view (e.g., the first key parameter to the first key element, the second key parameter to the second key element, etc.).

**Format** Expression of the same type and size as the corresponding key element in the external file view.

**FIRST**

**Description** *Optional.* Retrieves the first record in the external file.

**NEXT**

**Description** *Optional.* Retrieves the next record in the external file.

**Consideration** NEXT is the default option for the GET statement.

**EQUAL**

**Description** *Optional.* Retrieves a record only if the specified key parameters are equal to the corresponding key elements. MANTIS returns FOUND if the record exists; otherwise, it returns NOTFOUND.

**Considerations**

- When EQUAL is not specified, MANTIS will retrieve the next record in key sequence and return NEXT if it cannot find an equal key.
- If you issue GET...EQUAL, a subsequent GET...NEXT returns the next record (in key sequence), even if a NOTFOUND status was returned on the GET...EQUAL.

## ENQUEUE

**Description** *Optional.* Obtains a lock on the record. The lock will be released at the next commit point (e.g., at an explicit COMMIT statement, at an implicit commit during screen I/O, when the file is closed with a RELEASE statement, or when the program terminates with a CHAIN or returns to the main facility menu). The lock will also be released if the record is updated or deleted.

**Consideration** A lock is only obtained if MANTIS is running in shared mode. Refer to the *MANTIS for Windows Administration Guide*, P19-2304, for a description of the /SHARE command line parameter.

## LEVEL=*level-number*

**Description** *Optional.* Specifies which level of MANTIS array elements the fields in the record will be copied to.

**Default** LEVEL=1

**Format** Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding ACCESS statement.

**Example**

```
20 ACCESS RECORD("INDEX","SERENDIPITY",16)
30 SCREEN MAP("INDEX")
40 CONVERSE MAP
50 COUNTER=1
60 WHILE MAP<>"CANCEL" AND COUNTER<17
70 .GET RECORD LEVEL=COUNTER
80 END
```

## General considerations

- Before you can read a record from an external file, you must open it with an ACCESS statement.
- In MANTIS for the IBM mainframe, the ENQUEUE parameter of the GET statement is not allowed for external files, and the NEXT parameter is not allowed when a key is also specified for external files.
- For INDEXED files, the key you specify in the GET statement must correspond to the key element(s) you specified during External File View Design.
- For SEQUENTIAL files, the key is the Relative Byte Address (RBA). MANTIS returns the RBA of the retrieved record in the associated reference variable if one was defined during External File View Design.
- For NUMBERED files, the key is the Relative Record Number (RRN). The first record has an RRN of 1. MANTIS returns the RRN of the retrieved record in the associated reference variable defined during External File View Design.
- The status of the GET can be obtained by using the symbolic name *access-name* to invoke a built-in text function. The status can be one of the following values:

| | |
|---|---|
| FOUND | MANTIS successfully retrieved the record identified by the key you supplied. |
| NEXT | MANTIS retrieved the next record in sequence for a GET without a key, or MANTIS retrieved the next record in key sequence when it could not locate the supplied key or when a partial key was supplied. |
| END | MANTIS failed to retrieve the record because it reached the end of the file. |
| NOTFOUND | EQUAL was specified, and MANTIS could not find a record with an equal key. |
| HE0LD | MANTIS failed to retrieve the record because it was held by another user. |

The following values may also be returned if TRAP is in effect:

| | |
|---|---|
| LOCK | The password specified in the ACCESS statement is invalid, or the record is locked by another user or another view. |
| ERROR | A physical error occurred while retrieving the record. The RBA for a SEQUENTIAL file was not at a record boundary, or the RRN for a RELATIVE file specified a record number outside the file range. |
| DATA | The record retrieved contained invalid data. |

- ♦ If the GET is unsuccessful, you can use the HELP LAST command (in Program Design) to examine the system error message that may have been returned. This information is also available to MANTIS fault handling routines (see “SET” on page 202).

- ♦ Records locked using ENQUEUE are released by a COMMIT statement, by automatic COMMIT processing (if enabled), on the next terminal read, when the external file is closed by a RELEASE statement, or by program termination.

## HEAD

Use the HEAD statement to center a heading on the top line of the screen in scroll mode. Headings do not appear on a formatted screen. Headings appear in scroll mode when MANTIS executes a SHOW, WAIT, or OBTAIN statement.

**HEAD *heading***

### *heading*

| | |
|---|---|
| **Description** | *Required.* Specifies the heading to be displayed at the top of the screen. |
| **Format** | Text expression with a length in the range of 0 to (screen width minus 2) characters. |

**Example**

```
10 ENTRY BUZZ PHRASE GENERATOR
20 .DO SET UP VOCABULARY
30 .HEAD "BUZZ PHRASE GENERATOR"
40 .CLEAR
50 .SHOW "I WILL GENERATE A SCREEN FULL OF BUZZ"
55 .'"PHRASES EVERY TIME YOU** PRESS 'ENTER'. "
60 .'"WHEN YOU WANT TO STOP, ENTER 'CANCEL'."
70 .UNTIL KEY="CANCEL"
80 ..INDEX=1
90 ..UNTIL INDEX=22
100 ...A=INT(RND(10)+1)
110 ...B=INT(RND(10)+1)
120 ...C=INT(RND(10)+1)
130 ...SHOW FIRST(A)+" "+SECOND(B)+" "+NOUN(C)
140 ...INDEX=INDEX+1
150 ..END
160 ..WAIT
170 .END
180 .CHAIN "GAMES MENU"
190 EXIT
```

## General considerations

- ♦ You can specify only one heading for each screen displayed.
- ♦ The heading remains in effect until another HEAD statement is executed.
- ♦ HEAD with a NULL operand clears the current heading (if any).

## IF-ELSE-END

Use the IF-ELSE-END statements to execute one block of statements if a specified condition is TRUE, another block if the condition is FALSE.

**IF *expression***

**.**

**[*true-block*]**

**.**

**ELSE**

**.**

**[*false-block*]**

**.**

**END**

### *expression*

| | |
|---|---|
| **Description** | *Required.* Specifies the condition that determines whether true-block or false-block is executed. |
| **Format** | Arithmetic expression that evaluates to a nonzero value (TRUE) or zero (FALSE). |

### *true-block*

| | |
|---|---|
| **Description** | *Optional.* Identifies the block of statements you want to execute if expression is TRUE. |

### *false-block*

| | |
|---|---|
| **Description** | *Optional.* Identifies the block of statements you want to execute if expression is FALSE. |

## Example

```
70 WHILE ONE<>TWO AND ONE>ZERO AND TWO>ZERO
80 .IF ONE<TWO
90 ..ONE=ONE-INT(ONE/TWO)*TWO
100 .ELSE
110 ..TWO=TWO-INT(TWO/ONE)*ONE
120 .END
130 END
```

## General considerations

- If expression is TRUE, MANTIS executes the statements in true-block (if present) and resumes at the statement following END.
- If expression is FALSE, MANTIS executes the statements in false-block (if present) and resumes at the statement following END.

# INSERT

Use the INSERT statement to insert a new record into a MANTIS file or an external file.

## MANTIS file INSERT

**INSERT *file-name* [LEVEL=*level-number*]**

### *file-name*

| | |
|---|---|
| **Description** | *Required.* Specifies the symbolic name of the file in which you want to insert a record. |
| **Format** | MANTIS symbolic name defined in a previously executed FILE statement. |

### LEVEL=*level-number*

| | |
|---|---|
| **Description** | *Optional.* Specifies which level of MANTIS array elements contains the fields in the record you want to insert. |
| **Default** | LEVEL=1 |
| **Format** | Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding FILE statement. |

## Example

```
10 ENTRY INDEX
20 .FILE RECORD("INDEX","SERENDIPITY")
30 .SCREEN MAP("INDEX")
40 .GET RECORD
50 .WHILE RECORD<>"END AND MAP<>"CANCEL"
60 ..CONVERSE MAP
70 ..WHEN MAP="PF1"
80 ...INSERT RECORD
90 ..WHEN MAP="PF2"
100 ...DELETE RECORD
110 ..WHEN MAP="PF3"
120 ...UPDATE RECORD
130 ..END
140 ..GET RECORD
150 .END
160 EXIT
```

## General considerations

- You must execute a corresponding FILE statement before you can execute the INSERT statement.
- The contents of key data elements identify where the record is to be inserted.
- The status of the INSERT can be obtained by using the symbolic name *file-name* to invoke a built-in text function.  The status is a null text value if the INSERT was successful.  MANTIS may also return a status of LOCK or ERROR if TRAP is in effect.

  | | |
  |---|---|
  | LOCK | The password specified in the FILE statement for this file profile is not valid for deletions or insertions. |
  | ERROR | A physical error occurred while inserting the record, the disk was full, or the file contained a record with the same key as the record to be inserted. |

- MANTIS performs an implicit COMMIT on every terminal input operation, unless this functionality is disabled by the COMMIT OFF statement.

## External file INSERT

**INSERT *access-name* [LEVEL=*level-number*]**

### *access-name*

**Description** *Required.* Specifies the symbolic name of the external file in which you want to insert a record.

**Format** MANTIS symbolic name defined in a previously executed ACCESS statement.

**Consideration** If the file has been closed, it will be automatically reopened, but the position in the file will be lost. For example, if a GET...NEXT is subsequently executed, the first record in the file is returned regardless of which record was being accessed before the release.

### LEVEL=*level-number*

**Description** *Optional.* Specifies which level of MANTIS array elements contains the fields in the record you want to insert.

**Default** LEVEL=1

**Format** Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding ACCESS statement.

**Example**

```
20 .ACCESS RECORD("INDEX","SERENDIPITY",16)
30 .SCREEN MAP("INDEX")
40 .CONVERSE MAP
50 .COUNTER=1
60 .WHILE MAP<>"CANCEL" AND COUNTER<17
70 ..WHEN INDICATOR(COUNTER)="G"
80 ...GET RECORD LEVEL=COUNTER
90 ..WHEN INDICATOR(COUNTER)="D"
100 ...DELETE RECORD LEVEL=COUNTER
110 ..WHEN INDICATOR(COUNTER)="I"
120 ...INSERT RECORD LEVEL COUNTER
.
.
.
```

## General considerations

- ♦ You must execute a corresponding ACCESS statement before you can execute the INSERT statement.
- ♦ For INDEXED files, the contents of key data elements identify where the record is to be inserted.
- ♦ For SEQUENTIAL files, MANTIS inserts the record at the end of the file.
- ♦ For NUMBERED files, the Relative Record Number (RRN) contained in the associated reference variable (defined during External File View Design) identifies the record to be inserted. You are allowed to insert a record anywhere in a DOS file. If you insert a record that is not immediately following the current record with the highest number, dummy records are automatically created (containing unpredictable data) between the highest-numbered record and the one you specify. For example, if the highest-numbered record is 8 and you insert record 12, dummy records 9-11 are automatically created. You can insert a record anywhere into a DOS NUMBERED file, even over an existing record to replace it.
- ♦ The status of the INSERT can be obtained by using the symbolic name *access-name* to invoke a built-in text function. The status is a null text value if the INSERT was successful. MANTIS may also return a status of LOCK or ERROR if TRAP is in effect.

| | |
|---|---|
| LOCK | The password specified in the ACCESS statement for this file view is not valid for deletions or insertions. |
| ERROR | A physical error occurred while inserting the record, the disk was full, or an INDEXED or VSAM NUMBERED file contained a record with the same key or RRN as the record to be inserted. |

- ♦ MANTIS performs an implicit COMMIT on every terminal input operation, unless this functionality is disabled by the COMMIT OFF statement.
- ♦ When inserting a record into a file with multiple keys, a duplicate on any key will cause the INSERT to fail.

## INT

Use the INT (integer) function to return the integer value of an arithmetic expression, truncating fractions.

**INT(*a*)**

***a***

**Description** *Required.* Specifies the number whose integer value you want returned.

**Format** Valid arithmetic expression.

**Examples**

| | | |
|---|---|---|
| `INT(45)` | returns | `45` |
| `INT(45.5)` | returns | `45` |
| `INT(-45.5)` | returns | `-45` |
| `INT(-.5)` | returns | `(0)` |

## INTERFACE

Use the INTERFACE statement to define an interface for your program to access. MANTIS retrieves the interface profile from the library and defines MANTIS variables for each field in the interface profile.

**INTERFACE *interface-name* (*libname*,*password*[,PREFIX][,*levels*]),...**

### *interface-name*

**Description** *Required.* Specifies the symbolic name for the interface profile in subsequent CALL statements.

**Format** Unique MANTIS symbolic name.

**Considerations**

- ♦ No processing occurs if *interface-name* is already defined.
- ♦ The *interface-name* symbolic name returns the 8-character interface status of the last CALL to *interface-name*.
- ♦ The Status field is initialized to the empty string by MANTIS prior to each CALL. The interface program can specify a text value for this field as a part of normal CALL processing.
- ♦ If MANTIS detects an error while processing the CALL, it will return a value of ERROR in the Status field. The FSI function can be used to obtain a more precise indication of the type of error detected. See "CALL" on page 90 in this chapter for more information.

**Example**

```
20 INTERFACE MASTER("CUSTOMERS","ALIBABA",10)
.
.
.
60 CALL MASTER("GET",1234)LEVEL=2
SHOW MASTER
```

### ***libname***

| | |
|---|---|
| **Description** | *Required.* Specifies the name of the interface profile as it was saved during Interface Design. |
| **Format** | Text expression that evaluates to a library name in the format: [*user name*:]*interface-name*. |
| **Consideration** | If the interface profile is in your own library, you do not need to specify user name. |

### ***password***

| | |
|---|---|
| **Description** | *Required.* Specifies the password needed to access the interface. |
| **Format** | Text expression that evaluates to the password specified during Interface Design. |

### **PREFIX**

**Description** *Optional.* Causes MANTIS to put the prefix interface name on all variable names associated with this interface. For example, if the interface BOTTLES has a field named VOLUME, the following statement defines the corresponding variable BIN VOLUME to be defined in the work area:

```
10 INTERFACE BIN("BOTTLES","MAGIC",PREFIX)
```

## *levels*

**Description** *Optional.* Specifies the number of levels for this interface that you want to use in CALL statements.

**Format** Arithmetic expression that evaluates to a value in the range 1–255.

**Considerations**

- ♦ If you specify *levels*, you must specify LEVEL=*level-number* (unless you want the default LEVEL=1) in all CALL statements for the interface.
- ♦ If you do not specify *levels*, MANTIS defines a BIG, SMALL, or TEXT variable or array for each field in the interface profile according to the type and dimensions specified during Interface Design.
- ♦ If you specify *levels*, MANTIS adds an extra dimension to each field to allow a separate value to be stored at each level. MANTIS defines a one-dimensional array instead of a variable, a two-dimensional array instead of a one-dimensional array, and so on. The first dimension of each array is *levels*.

**Example**

```
20 INTERFACE MASTER("CUSTOMERS","ALIBABA",10)
.
.
.
60 CALL MASTER("GET",1234)LEVEL=2
```

# KEY

Use the KEY function to return a text value identifying your response to the most recent CONVERSE, OBTAIN, PROMPT, or WAIT statement. For CONVERSE, the value is the name of the action or macro key you entered (see the table in the “MANTIS action keys” section on page 35). For OBTAIN, PROMPT, and WAIT, this value is replaced by the contents of the Key Simulation field.

**KEY**

## Example

```
10 ENTRY BUZZ_PHRASE_GENERATOR
20 .DO SET_UP_VOCABULARY
30 .HEAD "BUZZ PHRASE GENERATOR"
40 .CLEAR
50 .SHOW "I WILL GENERATE A SCREEN FULL OF BUZZ"
55 ."'PHRASES EVERY TIME YOU PRESS 'ENTER'. "
60 ."'WHEN YOU WANT TO STOP, ENTER 'CANCEL'."
70 .UNTIL KEY="CANCEL"
80 ..INDEX=1
90 ..UNTIL IN²`DEX=22
100 ...A=INT(RND(10)+1)
110 ...B=INT(RND(10)+1)
120 ...C=INT(RND(10)+1)
130 ...SHOW FIRST(A)+""+SECOND(B)+""+NOUN(C)
140 ...INDEX=INDEX+1
150 ..END
160 ..WAIT
170 .END
180 .CHAIN "GAMES_MENU"
190 EXIT
```

## General considerations

- In MANTIS for the IBM mainframe, the KEY function always returns the actual key pressed.
- The value remains available if you CHAIN to another program or DO an external subroutine.

## LANGUAGE

Use the LANGUAGE function to return the default language of the user who is currently signed on.

**LANGUAGE**

### Example

```
SHOW LANGUAGE
```

### General consideration

The LANGUAGE function is not supported by MANTIS for the IBM mainframe.

# LET

Use the LET statement to assign the value of an expression to a variable, array element, or text substring, or to assign values to a sequence of consecutive array elements.

**[LET]** ***variable*** **[ROUNDED[(*places*)]]=*expression*,...**

### *variable*

| | |
|---|---|
| **Description** | *Required.* Specifies the variable or array element or text substring to which you want to assign a value. |
| **Format** | Name of a BIG, SMALL, or TEXT variable or the subscripted name of a BIG, SMALL, or TEXT array. |

### ROUNDED

| | |
|---|---|
| **Description** | *Optional.* Specifies whether the value of an arithmetic expression is to be rounded before MANTIS assigns it to the numeric variable or array element. |

### *places*

| | |
|---|---|
| **Description** | *Optional.* Specifies the number of decimal places to round the expression before MANTIS assigns it to the numeric variable or array element. |
| **Default** | 0 |
| **Format** | Integer in the range 0–6. |

### *expression*

| | |
|---|---|
| **Description** | *Required.* Specifies the value to be assigned to the variable or array element or text substring. |
| **Format** | Logical or arithmetic expression for assignment to a BIG or SMALL variable or array element; text expression for assignment to a text variable, array element, or substring. |

**Consideration** If you specify more than one expression, the variable must be a subscripted array name. MANTIS increments the array subscript(s) after assigning each value. The rightmost subscript is incremented first for a multidimensioned array. Whenever a subscript exceeds its dimension, it is set to 1, and the next subscript to the left is incremented.

**Examples**

```
LET ANSWER ROUNDED(2)=CAPITAL*(1+RATE/100)**LENGTH
```

is equivalent to:

```
ANSWER ROUNDED(2)=CAPITAL*(1+RATE/100)**LENGTH

A=(B+C)*D

BIG X
TEXT Y(16)
LET X=VALUE(Y)
LET Y=TXT(X)

X=X+1

BIG PRIMES(10)
PRIMES(1)=2,3,5,7,11,13,17,19,23,29

TEXT ANSWER(20)
ANSWER(1,13)="The answer is"
ANSWER(17)=TXT(42)
```

## General considerations

- If the symbolic name has not been defined on a BIG, SMALL, or TEXT statement, MANTIS implicitly defines it as a BIG (numeric) variable.
- Use the ROUNDED option when you want to control fractional results (e.g., in currency calculations).
- To assign a numeric value to a text variable or array element, you must convert it to a text value using the built-in function TXT (as in the example above).
- To assign a text value to a numeric variable or array element, you must convert it to a numeric value using the built-in function VALUE (as in the example above).
- A new text value can be assigned to a text substring by specifying one or two subscripts (see "Variables and arrays" on page 45):
    - If the first and last character positions are specified, the new value is truncated or padded with spaces to the same length as the substring. The length of the text variable remains unchanged unless the substring extends past the end of the old value.
    - If only the first character position is specified, the last position is effectively determined by the maximum length of the text variable, and the new value is truncated to the corresponding length. The length of the text variable depends on the length of the new value.
- In either case, any positions between the end of the old text value and the start of the substring are padded with spaces.

## LOG

Use the LOG function to return the natural logarithm of an arithmetic expression (inverse of the EXP function).

**LOG(*a*)**

***a***

**Description** *Required.* Specifies the number whose natural logarithm you want returned.

**Format** Valid arithmetic expression.

**Example**

```
LOG(1000) returns 6.90775527898
```

## MODIFIED

Use the MODIFIED function to determine how many fields were modified or whether a specific field on a screen was modified during the last CONVERSE statement.

**MODIFIED(*screen-name*[,*field-name*])**

### *screen-name*

| | |
|---|---|
| **Description** | *Required.* Specifies the symbolic name of the screen you are testing, as defined in a SCREEN statement. |
| **Format** | MANTIS symbolic name. |

### *field-name*

| | |
|---|---|
| **Description** | *Optional.* Specifies the symbolic name of the field you are testing, as defined in Screen Design. |
| **Format** | MANTIS symbolic name. |
| **Consideration** | If you supply a field name, MANTIS returns TRUE if you alter the specified field. Otherwise, MANTIS returns FALSE. |

## Example

```
10 SCREEN CLIENT_INFO("CLIENT_INFORMATION")
20 CONVERSE CLIENT_INFO
30 IF MODIFIED(CLIENT_INFO)
40 .DO CLIENT PROCESSING: ¦ ONE OR MORE FIELDS CHANGED
50 END
60 .
70 .
80 .
90 ENTRY CLIENT_PROCESSING
100 .FIELDS_TO_CHECK=MODIFIED(CLIENT_INFO)
110 .I=1
120 .LIMIT=SIZE(CLIENT_NAME,1)
130 .WHILE I<=LIMIT AND FIELDS_TO_CHECK
140 ..IF MODIFIED(CLIENT_INFO,CLIENT_NAME(I))
150 ... ¦ perform any edit checks
160 ...FIELDS_TO_CHECK=FIELDS_TO_CHECK-1
170 ..END
180 ..I=I+1
190 .END
200 EXIT
```

## General considerations

- MANTIS returns an implicit TRUE condition (IF MODIFIED(map)) for any value greater than zero. An explicit TRUE condition (IF MODIFIED (map1)=TRUE) occurs only if the function has a value of one.

- If you do not supply a field name, MANTIS returns the number of fields altered during the last terminal I/O. MANTIS returns 0 (FALSE) if no fields were modified.

- MANTIS resets modified field indicators only when the map set is displayed. In other words, MANTIS does not reset modified field indicators when you converse a screen with the WAIT option.

- Because zero evaluates to FALSE, you can use MODIFIED as a logical or arithmetic function.

# NEXT

Use the NEXT statement to transfer control to the next conditional repeat in a FOR-END, UNTIL-END, or WHILE-END statement or to the next WHEN condition in a WHEN-END statement.

**NEXT**

## Example

```
10 FOR L=1 TO MAXLINES : ¦ For each screen line
20 .IF NOT(MODIFIED(CUST_DETAILS,CUST_CREDIT(L)))
30 ..NEXT: ¦ Continue if customer credit not modified
40 .END
50 .UPDATE CUSTOMER LEVEL=L : ¦ Update customer record
60 END
```

## General considerations

- In FOR-END, UNTIL-END, and WHILE-END statements, NEXT results in the next repeat only if the respective condition is satisfied. If the condition is not satisfied, the statement after the END statement is executed next.
- In FOR-END statements, NEXT causes MANTIS to add the increment to the counter before comparing the counter with the final value.
- In WHEN-END statements, NEXT causes MANTIS to drop through to the next WHEN condition or to the END statement.
- With nested logic statements, NEXT proceeds with the innermost FOR-END, UNTIL-END, WHEN-END, or WHILE-END statements.

## NOT

Use the NOT function to return one (TRUE) if an arithmetic expression evaluates to zero (FALSE); otherwise, NOT returns zero.

**NOT(*a*)**

***a***

**Description** *Required.* Specifies the number you want evaluated to TRUE or FALSE.

**Format** Valid arithmetic expression.

**Example**

```
10 IF NOT(A=3 OR J=TRUE)
.
.
.
100 END
```

## NULL

Use the NULL function to return an empty (zero-length) text value.

**NULL**

**Example**

```
.IF CLIENT_NAME=NULL
..DO ERROR_PROCESSING
```

**General consideration**

NULL is supported by Releases 5.2 and above of MANTIS for the IBM mainframe.

## NUMERIC

Use the NUMERIC function to determine if a text expression contains a valid number.

**NUMERIC(*text-expression*)**

### *text-expression*

**Description** *Required.* Specifies the expression you want MANTIS to check.

**Format** Text expression.

**Examples**

| | | |
|---|---|---|
| `NUMERIC("123,456.789")` | returns | `TRUE` |
| `NUMERIC("$1234.55")` | returns | `FALSE` |
| `NUMERIC("-.05")` | returns | `TRUE` |
| `NUMERIC ("")` | returns | `FALSE` |
| `ABC="-1234.55"` `NUMERIC(ABC+"44")` | returns | `TRUE` |

**General considerations**

- ♦ TRUE is returned if the text expression contains a valid number; FALSE is returned if it is not a valid number.
- ♦ A valid number is defined by the following rules:
  - May contain one decimal point.*
  - May contain commas in valid positions.*
  - May have a leading or trailing sign character, but not both. The sign character may be separated from the rest of the number by blanks.
  - Must contain at least one number (0–9) without embedded spaces.
  - May not contain currency notation (e.g., $).
  - E notation is not allowed.

* The decimal point and comma are defined in your User Profile.

## OBTAIN

Use the OBTAIN statement to obtain data values from an unformatted screen and assign them to numeric or text variables or array elements. You can also use OBTAIN to obtain data entered in the Unsolicited Input field on a formatted screen after the last CONVERSE statement was executed.

**OBTAIN *variable*,...**

### *variable*

**Description** *Required.* Specifies a variable or array element for which you want to obtain a value.

**Format** Name of a BIG, SMALL, or TEXT variable or the subscripted name of a BIG, SMALL, or TEXT array.

**Consideration** You can mix different types of variables in an OBTAIN statement.

**Examples**

♦
```
 5 FILE RECORD("DEMO_LIST","PASS3")
10 SCREEN MAP("DEMO")
20 SHOW "ENTER KEY FOR INQUIRY:";
30 OBTAIN ACCT_NUMBER
```

♦
```
 5 SHOW "PLEASE ENTER MONTH;DAY;YEAR"
10 OBTAIN MONTH,DAY,YEAR
```

During execution of the last example, this would appear on the screen as follows:

```
PLEASE ENTER MONTH;DAY;YEAR
  4;2;89
```

## General considerations

- ♦ When you enter values in response to an OBTAIN statement, separate the values with a semicolon (;).
- ♦ If there has been no intervening SHOW statement since the most recent CONVERSE statement, the OBTAIN statement will obtain any data entered in the Unsolicited Input field on the bottom line of the screen. If no data was entered in the Unsolicited Input field, the variables in the OBTAIN statement will remain unchanged.
- ♦ If the OBTAIN statement follows a SHOW, MANTIS waits for you to enter a line of data followed by any action key. MANTIS assigns the values you enter to successive variables in the OBTAIN statement. If you enter more values than there are variables, the superfluous values are ignored. If you enter fewer values than there are variables, the variables for which there are no values remain unchanged.
- ♦ If there are data items pending from a previous SHOW statement ending with a comma (,) or a semicolon (;), the data items are displayed, and input is accepted from the following position on the screen.

## OUTPUT

Use the OUTPUT statement to route output from the CONVERSE and SHOW statements to the screen and/or printer.

$$\textbf{OUTPUT} \left\{ \begin{array}{l} \textbf{SCREEN} \\ \textbf{PRINTER[VIA exit]} \\ \textbf{SCREEN PRINTER [VIA exit]} \end{array} \right\}$$

### SCREEN

**Description** *Optional.* Sends output from CONVERSE or SHOW to your screen.

### PRINTER

**Description** *Optional.* Sends output from CONVERSE or SHOW to your printer.

**Consideration** If an OUTPUT statement specifies routing to the printer and the current program does not contain a PRINTER statement, MANTIS routes output using the default printer device name and spool command specified in your user profile.

### VIA exit

**Description** *Optional.* This parameter is for compatibility with MANTIS for the IBM mainframe and is ignored by MANTIS for Windows.

**Examples**

- The following example routes output to your screen (MANTIS normally routes output to the screen unless you specify otherwise):

  ```
  OUTPUT SCREEN
  ```

- The following example routes output from SHOW and CONVERSE statements to your printer. You can print a listing of the current program by using the OUTPUT PRINTER statement in immediate mode followed by a LIST command. After MANTIS has printed the listing, output routing reverts to your screen.

  ```
  OUTPUT PRINTER
  ```

- The following example routes output to both your screen and printer:

  ```
  OUTPUT SCREEN PRINTER
  ```

## General considerations

- MANTIS for the IBM mainframe gives control to a printer exit if the VIA exit parameter of the OUTPUT statement is used. In MANTIS for Windows, the VIA exit parameter is ignored.
- The OUTPUT statement clears all output lines from the SHOW statement that WAIT or OBTAIN have not forced out to the terminal.
- When OUTPUT PRINTER is in effect but OUTPUT SCREEN is not, the WAIT, OBTAIN, and CONVERSE statements simulate the condition of the ENTER key being pressed. MANTIS does not suspend the program.
- The print file is closed when an OUTPUT SCREEN statement is executed, when control returns to the user's facility program, or when a PRINTER statement is executed. When the print file is closed, spooling may occur (see "ATTRIBUTE" on page 81 and "PRINTER" on page 182 ).
- Different output is obtained in the Print File depending on whether you use OUTPUT PRINTER or OUTPUT SCREEN PRINTER. With OUTPUT PRINTER, the window size is the size of the current print or class. With the OUTPUT SCREEN PRINTER, the size of the terminal screen (default 24, 80) is the size of the window. What you see on your physical screen is what is printed.

## PAD

Use the PAD statement to insert padding characters into a text variable, array element, or substring.

$$\textbf{PAD}\ \textit{\textbf{string}}\left[\textit{\textbf{padding}}\right]\left[\begin{array}{l}\textbf{BEFORE}\\ \underline{\textbf{AFTER}}\\ \textbf{ALL}\end{array}\right]$$

### *string*

| | |
|---|---|
| **Description** | *Required.* Specifies the text variable, array element, or substring which you want to pad with padding characters. |
| **Format** | Symbolic name of a TEXT variable, the subscripted name of a TEXT array or a reference to a TEXT substring. |

### *padding*

| | |
|---|---|
| **Description** | *Optional.* Specifies the padding character. |
| **Default** | (blank) |
| **Format** | Text expression. |
| **Consideration** | The first character is the padding character; any subsequent characters are ignored. |

### BEFORE

| | |
|---|---|
| **Description** | *Optional.* Inserts padding characters before the first character in the string (the leading pad character). |

### AFTER

| | |
|---|---|
| **Description** | *Optional.* Inserts padding characters after the last character in the string (the trailing pad character). |
| **Default** | After. |

## ALL

**Description** *Optional.* Inserts padding characters before and after the current characters in the string (the leading and trailing pad characters).

**Consideration** If you specify ALL, MANTIS centers the current value within the padding characters.

**Examples**

- Center a title using the asterisk (*):

```
TEXT A(24)
"CLIENT NAME"                 "*******CLIENT NAME******"
PAD A "*" ALL
```

- Right justify using blanks:

```
TEXT A(24)
A=CLIENT NAME                   "             JOE JACKSON"
PAD A BEFORE
```

- Initialize a field to blanks or any character:

```
FILL CHAR=*
TEXT A,B
PAD A                                    "                "
PAD B FILL CHAR                "****************"
```

- Set the middle of a string to a character:

```
PAD CLIENT NUMBER(9,12)"*"   "12345678****DEFG"
```

## General considerations

- When the string is not a substring, the PAD statement sets the text variable or array element to its maximum length by inserting padding characters before or after the current text value.  PAD has no effect if the current text value is already at the maximum length.
- You cannot use BEFORE, AFTER, or ALL when padding a substring.
- You can pad a substring by specifying one or two subscripts (see “Variables and arrays” on page 45):
  - If the first and last character positions are specified, the length of the text variable remains unchanged unless the substring extends past the end of the old value.
  - If only the first character position is specified, the last position is effectively determined by the maximum length of the text variable.  The length of the text variable becomes the maximum length.
  - In either case, the padding character is inserted in each position of the corresponding substring.  Any positions between the end of the old text value and the start of the substring are padded with spaces.

# PASSWORD

Use the PASSWORD function to return a text value containing the password entered during MANTIS sign-on.

**PASSWORD**

### Example

```
10 ENTRY CUST ENTRY
20 .SCREEN MAP("CUST ENTRY")
30 .FILE REC("CUST FILE",PASSWORD)
40 .FILE RECX("CUST FILE",PASSWORD,PREFIX)
50 .CONVERSE MAP
60 .WHILE MAP<>"CANCEL"
.
.
.
```

### General consideration

PASSWORD always returns a text string of 1–16 characters.

## PERFORM

Use the PERFORM statement to invoke a DOS command.  The *command*, which can run a user-written program, executes without accessing or interfering with the MANTIS work area.

**PERFORM *command***

### *command*

**Description** *Required.*  Specifies the *command* to be invoked.

**Format** Text expression that evaluates to a DOS command.

**Example**

```
10 ENTRY FACILITY
 20 .SCREEN MAP("MENU")
30 .CONVERSE MAP
40 .WHILE MAP<>"CANCEL"
50 ..WHEN OPTION=1 OR MAP="PF1"
60 ...CHAIN "CUSTOMER NAMES"
70 ..WHEN OPTION=2 OR MAP="PF2"
80 ...PERFORM "INVOICES"
.
.
```

## General considerations

- In MANTIS for Windows, the parameter of the PERFORM statement is a DOS command that can perform any function, including running a program. In MANTIS for the IBM mainframe, the parameter of the PERFORM statement is the name of the program or transaction to be performed (with other optional information).

- After issuing a PERFORM statement, non-MANTIS output may remain on your screen. In this case, pressing SELECT will not copy what you see on your screen. To see what you are copying from the scroll map, press REFRESH before pressing SELECT.

- A shorthand form of the PERFORM statement is available for immediate-mode execution in programming mode. Entering a dollar sign ($) at the start of the line is equivalent to PERFORM. (This abbreviation is not supported by MANTIS for the IBM mainframe.) However, you cannot use a dollar sign ($) instead of the PERFORM statement in a MANTIS program. The command after $ is a text value instead of a text expression, so that the following examples in pairs are equivalent:

  ```
  $CHKDSK
  PERFORM "CHKDSK"
  $RENAME FILE1.DAT FILE2.DAT
  PERFORM "RENAME FILE1.DAT FILE2.DAT"
  ```

- You cannot PERFORM MANTIS.

- You can use PERFORM to build menu-driven systems where the menu itself and some system components are written in MANTIS. From MANTIS you can use the PERFORM statement to invoke existing or performance-sensitive components written in COBOL, BASIC, Pascal, Assembler, C, and so on.

## PI

Use the PI function to return the value of pi (3.14159265359).

**PI**

### Example

```
90 Y=100
100 X=PI*Y
110 SHOW X
.
.
```

## POINT

Use the POINT function to return the character position where joining or removal occurs in a text expression.

**POINT(*t*±*t*)**

***t***

**Description** *Required.* Specifies a text expression for which you want the joining or removal character positions returned.

**Format** Valid text expression.

**Example**

```
10 TEXT CUST_NO(11),DASH(1)
20 CUST_NO="222-22-2222":DASH="-"
30 A=POINT(CUST_NO-DASH)
```

Statement 30 places the value 4 in the variable A because the dash appears in the fourth position of CUST_NO.

**General consideration**

You can use POINT to check for an occurrence of a specific character string. For example, if you want those parts (PART_NAME) that contain the string “ECB” in their names, you could enter:

```
IF POINT(PART NAME-"ECB")
```

This expression returns TRUE if there is a match, or FALSE if there is not a match.

## PRINTER

Use the PRINTER function to invoke a built-in text function that returns the current printer assignment in the format *filename*;*spool-command*, described in the following list:

- *filename*   Specifies the name of the device to be assigned as the current printer, or the name of a file to be produced.
- *spool-command*   Specifies a DOS command to be issued when the file is closed.

For more information on printer assignments, see the considerations under the “PRINTER=” statement on page 183.

**PRINTER**

### Example

```
SHOW PRINTER
```

## PRINTER=

Use the PRINTER= statement to direct MANTIS printer output to a file or device. Any characteristics specified in Screen Design (such as color) that are not supported by the printer are ignored. Some attributes (such as BRIGHT and UNDERLINE) can be simulated (e.g., using bold printing for BRIGHT fields).

**PRINTER=*printer-spec***

### *printer-spec*

**Description** *Required.* Specifies the name of the device to be assigned as the current printer or the name of a file to be produced. You can also specify a system command to be issued when the file is closed.

**Format** Text expression that evaluates to [*filename*][;[*spool-command*]].

**Considerations**

- ♦ If you omit the file name or semicolon and spool command, the current file name or current spool command remains in effect.
- ♦ When the printer is closed, each occurrence of the hash character (#) in the spool command is replaced with the printer file name and the resulting command is performed as a DOS command.
- ♦ To erase the current spool command, use the semicolon (;) without the spool command (or with a blank spool command). You can also disable spooling by specifying ATTRIBUTE(PRINTER)=“NOS”.
- ♦ The spool command is only used for printer output to a file. It is ignored for devices (e.g., “PRN”).

## Examples

- PRINTER="MANTIS.LST;PRINT #" means *filename* is set to MANTIS.LST and *spool-command* is set to "PRINT #"
- PRINTERPRINTER="MANTIS.LST" means *filename* is set to MANTIS.LST and *spool-command* remains unchanged from previous PRINTER statement.
- PRINTER="MANTIS.LST;" means *filename* is set to MANTIS.LST and *spool-command* is set to null
- PRINTER=";PRINT #" means *filename* remains unchanged from the previous PRINTER statement and *spool-command* is set to "PRINT #".
- PRINTER=";" means *filename* remains unchanged from previous PRINTER statement and *spool-command* is set to null.
- The following examples are equivalent (they assign a physical device to receive printer output):

```
PRINTER="PRN"
10 TEXT DEVICE(4)
20 DEVICE="PRN"
30 PRINTER=DEVICE
```

## General considerations

- The PRINTER statement closes any open print file prior to processing the new printer specification and resets the output to OUTPUT SCREEN.
- In MANTIS for Windows, the parameter of the PRINTER statement is either the device name of the printer or a file name to which printed output will be written. In MANTIS for the IBM mainframe, the parameter of the PRINTER statement is a CICS or VTAM identifier. These parameters are not meaningful in programs migrated from Windows to the IBM mainframe. We recommend that applications to be migrated from Windows to the IBM mainframe do not use the PRINTER statement and send their printed output to the default printer defined in the user profile.
- If an OUTPUT statement specifies routing to the printer and the current program does not contain a PRINTER statement, MANTIS routes output using the default printer device name and spool command specified in your user profile.
- Printer CLASS specifies how data is sent to a printer. You can change the printer CLASS by using the ATTRIBUTE statement. You can also use the ATTRIBUTE statement to override the page size for the printer.

## PROGFREE

Use the PROGFREE function to return the number of bytes remaining in the program work area.

**PROGFREE**

### Example

```
SHOW PROGFREE
```

## PROGRAM

Use the PROGRAM statement to define an external subroutine to be invoked by a DO statement.

**PROGRAM *program-name*(*libname*,*password*),...**

### *program-name*

**Description** *Required.* Defines the symbolic name you will use when referring to the external subroutine in subsequent DO statements.

**Format** Unique MANTIS symbolic name.

### *libname*

**Description** *Required.* Specifies the name of the program as you saved it in Program Design.

**Format** Text expression that evaluates to [*user name*:]*program name*.

**Consideration** If the program is in your own library, you do not need to specify user name.

### *password*

**Description** *Required.* Specifies the password used when the program was saved during Program Design.

**Format** Text expression that evaluates to the password with which the program was saved or replaced (or the default password if none was specified).

**Consideration** If you supply an incorrect password to a program in another user's library, you will still be allowed to execute the external subroutine, but you will not be allowed to list or edit it if it stops (because of a fault condition) in programming mode.

### Example

```
100 ENTRY EDIT_PROGRAM
.
.
.
150 .PROGRAM EDIT_RTN("VALIDATION","COMMON")
160  .TYPE="CREDIT CHECK"
170 .DO EDIT_RTN(TYPE,CUST_NO,STATUS,MESSAGE)
180 .IF STATUS<>"GOOD"
190 ..DO ERROR_RTN(CUST_NO)
200 .END
210 .TYPE="SELECT SALES REP"
220 .DO EDIT_RTN(TYPE,CUST_NO,STATUS,MESSAGE)
230 .IF STATUS<>"GOOD"
240 ..DO ERROR_RTN(SALES_REP)
250 .ELSE
260 ..SALES_REP=MESSAGE
270 .END
280 EXIT
```

### General considerations

- The *libname* parameter identifies a MANTIS program that must begin with a subroutine defined by ENTRY-EXIT statements.  Only the first subroutine is accessible directly from the calling program, and any subsequent subroutines are internal to the external subroutine's program context.

- You must place an ENTRY-EXIT statement pair around the first subroutine in the external program.  The entry name in the ENTRY statement is not used.

- Statements such as SCREEN, FILE, ACCESS, and INTERFACE, that can generate a large amount of processing, should not be included in low-level external subroutines where they would be reexecuted on each DO.  If possible, insert them in a high-level routine and pass the associated symbolic names as arguments to low-level external subroutines.

- See "External DO" on page 267 for additional information on the PROGRAM statement as it works with external DO.

## PROMPT

Use the PROMPT statement to display a prompter. For chained prompters, MANTIS displays each prompter in the chain. Following the PROMPT, MANTIS returns control to the next line in the program.

**PROMPT *libname***

### *libname*

**Description** *Required.* Specifies the name of the prompter as you saved it in Prompter Design.

**Format** Text expression that evaluates to [user name:]prompter name.

**Consideration** If the prompter is in your own library, you do not need to specify user name.

**Example**

```
.
100 IF REQUEST="?"
120 .PROMPT "MASTER:FACILITY_HELP"
140 END
.
```

**General considerations**

- Each prompter can be a maximum of 80 lines. You can chain prompters to display more lines than with a single PROMPT statement. (For more information on designing prompters, refer to the *MANTIS for Windows Facilities Reference Manual*, P19-2301).
- Press ENTER to view the next screen in the prompter. When prompter information ends, MANTIS returns control to the next line in the program. You can terminate the PROMPT statement with the KILL command. Terminate subsequent chained prompters by pressing CANCEL.

## RELEASE (statement)

Use the RELEASE statement to close an external file, release internal storage used by an external subroutine, or close an interface.

$$\textbf{RELEASE}\left\{\begin{array}{l}\textbf{\textit{access - name}}\\\textbf{\textit{interface - name}}\\\textbf{\textit{program - name}}\end{array}\right\}$$

### *access-name*

**Restriction** The *access-name* parameter is not supported by MANTIS for the IBM mainframe.

**Description** *Optional.* Specifies the name of the external file you want to close.

**Format** MANTIS symbolic name defined in a previously executed ACCESS statement.

**Consideration** The specified external file is closed so that your program will not prevent another program, such as a PERFORMed program, from opening it. The current position in the file is lost. If you execute a GET, UPDATE, INSERT, or DELETE on a closed external file, however, MANTIS automatically reopens it. It is not necessary to reexecute the ACCESS statement. If you do not use RELEASE to close an external file, it will remain open until your program or subroutine terminates.

## *interface-name*

**Restriction** The *interface-name* parameter is not supported by MANTIS for the IBM mainframe.

**Description** *Optional.* Specifies the name of the interface you want to close.

**Format** MANTIS symbolic name defined in a previously executed INTERFACE statement.

**Considerations**

- The specified interface is closed so that your program will not prevent another program, such as a MANTIS program in another session (or in a network), from opening it and using it.
- For PROGRAM interfaces, the interface program will be terminated (if it is still running) and deleted from memory.
- For DLL interfaces, the DLL will be released from memory (unless it is in use by another process).
- If additional REMOTE interfaces in the same MANTIS session use the same pipe, they are all affected by a single RELEASE statement, because MANTIS will close that pipe.
- If you execute a CALL statement on an interface that you previously closed (via RELEASE), it will be reopened. It is not necessary to reexecute the INTERFACE statement.

## *program-name*

**Description** *Optional.* Specifies the name of the external subroutine that you want to remove from internal storage.

**Format** MANTIS symbolic name defined in a previously executed PROGRAM statement.

**Consideration** After you execute an external subroutine for the first time, MANTIS keeps it in internal storage so that it does not need to be reloaded if you execute it again. MANTIS may run more efficiently if you use RELEASE to release the internal storage when the subroutine is no longer needed.

## Example

```
210 ACCESS RECORD("INDEX","SERENDIPITY",16)
220 .
230 .
240 RELEASE RECORD
310 PROGRAM EDIT_RTN("VALIDATION","COMMON")
320 .
330 .
340 DO EDIT_RTN(TYPE,CUST_NO,STATUS,MESSAGE)
350 .
360 .
370 RELEASE EDIT_RTN
```

## General considerations

- See “External DO” on page 267 for additional information on the RELEASE statement and how it works with external DO.

## RELEASE (function)

This is a function;  the previous discussion was for the RELEASE statement.

Use the RELEASE function to obtain the current release and version number of MANTIS and the operating environment under which MANTIS is running.

**RELEASE**

The release function returns a string as follows:

```
vvvv = ssssssssss to ccccc .....
```

where:

*vvvv* is the release and version number of MANTIS .  The first two characters are the major and minor parts of the release number (e.g., “21” for MANTIS 2.1) and the last two characters are used to differentiate maintenance levels.

*ssssssssss* is the operation system environment and will be Windows.

*ccccc* ..... is a copyright message.

## RESET

Use the RESET statement to back out a Logical Unit of Work (LUW). Database updates are rolled-back to the most recent COMMIT point. RESET has no effect for the MANTIS cluster or for external PC files.

**RESET**

### Example

```
.
.
.
100 FILE REC("EXAMPLES:CUSTOMERS","CASINO")
.
.
.
160 TRAP REC ON
170 INSERT REC
.
.
.
220 UPDATE REC
230 IF REC="ERROR"
240 .RESET
250 END
.
.
.
```

# RETURN

Use the RETURN statement to return control from a subroutine or stop execution of a program.

**RETURN**

## Example

```
10 ENTRY BROWSE
20 .SCREEN MAP1("INDEX")
30 .FILE REC1("INDEX","SERENDIPITY")
40 .GET REC1
50 .WHILE REC1="NEXT"
60 ..CONVERSE MAP1
70 ..IF MAP1="CANCEL"
80 ...RETURN
90 ..END
100 ..GET REC1
110 .END
120 EXIT
```

## General considerations

- RETURN is not supported by MANTIS for the IBM mainframe.
- Use the RETURN statement when you want a subroutine to return control from any line within the ENTRY-EXIT statements. MANTIS returns control immediately as if the EXIT statement had been executed.
- If you use a RETURN statement in your main program, the effect is the same as a STOP statement.

## RND

Use the RND function to return a random real number between zero and a specified number, excluding zero and that number.

**RND(*a*)**

***a***

**Description** *Required.* Specifies the number you want as one end of the range (zero is the other end of the range).

**Format** Valid arithmetic expression.

**Consideration** Neither zero nor *a* will be generated.

**Example**

```
10 ENTRY BUZZ PHRASE GENERATOR
20 .DO SET_UP_VOCABULARY
30 .HEAD "BUZZ PHRASE GENERATOR"
40 .CLEAR
50 .SHOW "I WILL GENERATE BUZZ PHRASES EACH TIME"
60 .SHOW "YOU HIT ENTER. ENTER 'END' TO STOP"
70 .OBTAIN ANSWER
80 .UNTIL ANSWER="END"
90 ..X=1
100 ..WHILE X<22
110 ...A=INT(RND(10)+1)
120 ...B=INT(RND(10)+1)
130 ...C=INT(RND(10)+1)
140 ...SHOW FIRST(A)+" "+SECOND(B)+" "+THIRD(C)
150 ...X=X+1
.
.
.
```

# SCREEN

Use the SCREEN statement to define a screen design you will use in your program.

**SCREEN *screen-name*(*libname*[,PREFIX]),...**

## *screen-name*

**Description** *Required.* Defines the symbolic name you will use when referring to the screen in subsequent CONVERSE statements.

**Format** Unique MANTIS symbolic name.

**Considerations**

- No processing occurs if *screen-name* is already defined.
- The *screen-name* symbolic name returns a text value identifying the last key you pressed (or entered in the Key Simulation field) in response to the most recent CONVERSE statement.

**Example**

```
20 .SCREEN MAP("INDEX")
30 .CONVERSE MAP
40 .WHILE MAP"CANCEL"
.
.
.
80 ..CONVERSE MAP
90 .END
```

## *libname*

**Description** *Required.* Specifies the name of the screen design as it was saved during Screen Design.

**Format** Text expression that evaluates to [*user name*:]*screen-name*.

**Consideration** If the screen design is in your own library, you do not need to specify user name.

## PREFIX

### Description

*Optional.* Specifies the prefix *screen-name* for all variable names associated with this screen.

For example, if the MANTIS screen BOTTLES has a field named VOLUME, the following statement causes the corresponding variable BIN VOLUME to be defined in the work area:

```
10 SCREEN BIN("BOTTLES",PREFIX)
```

### Example

```
20 .FILE RECORD("INDEX","SERENDIPITY",16)
30 .SCREEN MAP("INDEX")
40 .WHILE RECORD<>"END" AND MAP<>"CANCEL"
50 ..CLEAR MAP:LEVEL_NUMBER=1
70 ..GET RECORD("WILLIAMS") LEVEL=LEVEL_NUMBER
80 ..WHILE RECORD<>"END" AND LEVEL_NUMBER<16
90 ...LEVEL_NUMBER=LEVEL_NUMBER+1
100 ...GET RECORD LEVEL=LEVEL_NUMBER
110 ..END
120 ..CONVERSE MAP
130 .END
```

### General consideration

If you are using automatic mapping between screens and files or views, you may want to code your SCREEN statements after your ACCESS or FILE statements. This ensures you are using the most currently defined value for the field. If one field is defined in two different ways, MANTIS uses the first definition it encounters in your program. Also, MANTIS automatically defines numeric fields in screen designs as BIG.

## SCROLL

Use the SCROLL statement to specify the amount by which windowing keys will move the window.

$$\textbf{SCROLL}\begin{bmatrix}\underline{\textbf{ON}}\\ \textbf{OFF}\\ \left[\boldsymbol{\mathit{row1}}\right]\left[\boldsymbol{,\mathit{col1}}\right]\end{bmatrix}$$

### ON

**Description** *Optional.* Provides for compatibility with MANTIS for the IBM mainframe which allows alternate scrolling modes. MANTIS for Windows ignores this parameter.

### OFF

**Description** *Optional.* Provides for compatibility with MANTIS for the IBM mainframe which allows alternate scrolling modes. MANTIS for Windows ignores this parameter.

### *row1*

**Description** *Optional.* Specifies the row increment for window movement. This is the number of rows that the window will move up or down in the logical display when you press WINUP/WINDOWN.

**Format** Arithmetic expression.

### *col1*

**Description** *Optional.* Specifies the column increment for window movement. This is the number of columns that the window will move left or right in the logical display when you press WINLEFT/WINRIGHT.

**Format** Arithmetic expression.

**Example**

```
10 SCREEN S("BIG SCREEN")
20 SCROLL 11,40
30 CONVERSE S
```

## General considerations

- The SCROLL statement with ON, OFF, or no parameter has no effect and is included only for compatibility with MANTIS for the IBM mainframe, which has alternate scrolling modes.  MANTIS for Windows always scrolls the screen when in scroll mode.  Each new line from a SHOW statement is displayed at the bottom of the screen after scrolling up the screen contents by one line.

- Use SCROLL with *row1* and *col1* parameters to specify window movement increments from your program.  You can also specify increments from the keyboard (preceded by *i*) in the row and column coordinates.

# SEED

Use the SEED statement to cause the random number generator to generate a new sequence of random numbers.

**SEED**

### Example

```
10 SEED
20 A=10
30 B=RND(A)
```

### General consideration

Without the SEED statement, the random number generator produces the same sequence of numbers each time you execute a program.

## SET

Use the SET statement to set a fault trap or assign a value to a system environment variable. Setting a fault trap causes a specified subroutine to be executed if a program fault occurs.

$$\textbf{SET}\left\{\begin{array}{l}\textbf{TRAP}\quad\left\{\begin{array}{l}\textbf{\textit{entry - name}}\\ \textbf{\textit{program - name}}\end{array}\right\}\\ \textbf{\$SYMBOL }\textbf{\textit{name}}\textbf{ TO }\textbf{\textit{value}}\end{array}\right\}$$

### TRAP *entry-name*

**Description** *Optional.* Specifies a MANTIS subroutine to take control when a MANTIS fault occurs. The subroutine is called a fault handler, and once invoked, it must either handle the error (possibly ignoring it), or stop the program.

**Format** A MANTIS symbolic name defined in an ENTRY statement.

### TRAP *program-name*

**Description** *Optional.* Specifies the name of the external subroutine to be executed if a program fault occurs.

**Format** MANTIS symbolic name defined in a previously executed PROGRAM statement.

### Considerations

- You can declare a fault handler at any MANTIS DOLEVEL. When a MANTIS fault occurs, the most recently declared handler is invoked (the handler at the highest DOLEVEL). When the handler EXITs or RETURNs, control resumes after the MANTIS program line in error, and always in the program context in which the fault handler was declared. This means that if a fault handler is declared in a MANTIS program, and an error occurs in an external subroutine, program control will resume in the calling program after the DO statement that invoked the faulty subroutine.

- A fault trap subroutine can obtain information about the fault by specifying up to four arguments on the ENTRY statement. The optional arguments contain the following text values:

  ```
  ENTRY handler-name [(p1 [,p2 [,p3 [,p4]]])]
  ```

  p1 Text parameter containing the MANTIS fault message. The first three characters reveal the MANTIS fault code. MANTIS fault codes are documented in "Messages" on page 299.

  p2 Text parameter containing the MANTIS line in error. For a bound program, the text of the program line is unavailable, and this argument will be set to "At line *nnnnn*". For an unbound program, the MANTIS line number and text are passed.

  p3 Text parameter containing the last error message saved by MANTIS, if any. This parameter shows what would otherwise be returned by the HELP LAST command.

  p4 Text parameter containing the name of the program in error. This is useful when the fault handler is at a lower DOLEVEL than the program causing the error. The name passed in this parameter is the same as that specified in the PROGRAM or CHAIN statement, or the RUN command, whichever was used to access the program originally.

- A SET TRAP can be reset with a CLEAR TRAP statement.

## $SYMBOL *name*

**Description** *Optional.* Specifies the name of an environment variable to which you want to assign a value.

**Format** Text expression that evaluates to a valid environment variable name.

## TO *value*

**Description** *Optional.* Specifies the value that you want to assign to the environment variable.

**Format** Text or arithmetic expression.

**Consideration** If the value parameter is an arithmetic expression, it is converted to a text value.

## Examples

```
10 SET TRAP FAULT
20 ACCESS F("FILE1","READ",FILE "dummy.dat")
30 GET F FIRST
40 STOP
50 ENTRY FAULT(ARG1,ARG2,ARG3)
60 SHOW "ARG1:";ARG1
70 SHOW "ARG2:";ARG2
80 SHOW "ARG3:";ARG3
90 EXIT
RUN
ARG1: 213 A MANTIS file or external file could not be opened
ARG2: 30 GET F FIRST
ARG3: 89/04/22 12:20:54 I/O Error: No such file or directory
  (error code=2)
```

```
SET $SYMBOL "MONTH" TO "NOVEMBER"
SHOW $SYMBOL("MONTH")
NOVEMBER
```

## General considerations

- SET is not supported by MANTIS for the IBM mainframe.
- SET $SYMBOL performs an environment variable string assignment. Note that because MANTIS works with a copy of the original environment, any changes made to environment variables will be lost when MANTIS terminates.

## SGN

Use the SGN (sign) function to return -1 if an arithmetic expression is less than 0, 0 if it equals 0, and +1 if it is greater than 0.

**SGN(*a*)**

***a***

| | |
|---|---|
| **Description** | *Required.* Specifies the number whose sign you want to determine. |
| **Format** | Valid arithmetic expression. |

**Examples**

| | | |
|---|---|---|
| `SGN(-14)` | returns | `-1` |
| `SGN(.00001)` | returns | `+1` |
| `SGN(0)` | returns | `0` |
| `SGN(14E-17)` | returns | `1` |

## SHOW

Use the SHOW statement to display data on the screen in scroll mode. You can specify data items that are to be displayed line by line with certain column position options. You can also use SHOW to display the fault trap subroutine.

```
       ┌                ┐
       │ data - item    │
       │ ,              │
SHOW   │ ;              │ ...
       │ AT column      │
       │ TRAP           │
       └                ┘
```

### *data-item*

**Description** *Optional.* Specifies a data item you want to display. MANTIS inserts the data item at the current column position and advances the current column position past the data item.

**Format** Arithmetic or text expression.

**Consideration** In compatibility mode, data items must be separated by a comma (,) or semicolon (;). When not in compatibility mode, the separator can be omitted to display the data items without intervening spaces.

### ,

**Description** *Optional.* Advances the current column position to the next screen zone. Screen zones are 13 characters wide and start in columns 1, 14, 27, and so on.

### ;

**Description** *Optional.* Advances the current column position by one space.

## AT *column*

**Description** *Optional.* Advances the current column position to the specified column number.

**Format** Arithmetic expression that evaluates to a valid column number.

**Considerations**

- ♦ If the specified column number was less than the current column position, MANTIS terminates the current line and starts a new line.
- ♦ In compatibility mode, a comma or semicolon must be used as a separator after AT column, but does not affect the column position. When not in compatibility mode, a comma or semicolon after AT column advances to the next screen zone or column position.

## TRAP

**Description** *Optional.* Display the name of the fault trap subroutine, if any, for the current program or external subroutine.

**Example**

```
100 SHOW "ENTER YOUR AGE";
110 OBTAIN AGE
RUN
ENTER YOUR AGE (input is obtained from here)
```

**General considerations**

- ♦ A shorthand form of the SHOW statement is available. Entering a colon (:) at the start of the line is equivalent to SHOW. MANTIS will change the colon (:) to the word SHOW automatically.
- ♦ For scroll mode output, MANTIS keeps track of the current column position which is where the first character of the next data item will be inserted. MANTIS updates the current column position whenever a data item is inserted or a new column position is specified.
- ♦ When a SHOW statement ends with an item other than a comma or semicolon, MANTIS terminates the current line and starts a new line. When a SHOW statement ends with a comma or semicolon, the next SHOW statement continues at the current column position.

- A SHOW statement with no parameters displays a blank line.
- After displaying a screenful of lines, MANTIS displays MORE in the Key Simulation field and waits for you to reply before displaying any more lines. This prevents lines from being scrolled off the screen before you have had time to read them.
- When an OBTAIN statement is executed, any pending lines from previous SHOW statements are displayed. If there is an unterminated line, MANTIS positions the cursor after the current column position. This enables you to use a SHOW statement to display a prompt message at the bottom of the screen.
- When a CONVERSE statement is executed, any terminated pending lines from previous SHOW statements are displayed, but are then overwritten by the full-screen mode display. If there is an unterminated line, it is displayed at the bottom of the screen. This enables you to use a SHOW statement to display data in the Message field.
- When a PROMPT statement is executed, any terminated pending lines from previous SHOW statements are displayed, but are then scrolled off the screen by the prompter. Any unterminated line is displayed before the first line of the prompter.
- PROMPT and CONVERSE completely overwrite the screen. You may need to precede these statements with a WAIT or OBTAIN statement to avoid overwriting lines displayed by previous SHOW statements before you have had time to read them. If your program returns to scroll mode by executing a SHOW statement after CONVERSE or PROMPT, the lines displayed by previous SHOW statements will again be visible on the screen or in the Unsolicited Input field.
- MANTIS displays numbers in standard decimal notation or in scientific notation, depending on their size (see “Numeric considerations” on page 54). Whichever notation is used, numbers never exceed 13 characters, the width of a screen zone.

## SIN

Use the SIN function to return the sine of an arithmetic expression in radians.

**SIN(*a*)**

***a***

**Description** *Required.* Specifies the number whose sine you want returned.

**Format** Valid arithmetic expression.

**Example**

```
SIN(100)  returns    -.5063656411
```

## SIZE

Use the SIZE function to return the size and dimensions of an expression, variable or array.

$$\textbf{SIZE}\left(\left\{\begin{array}{ll} \textbf{\textit{text - expression}} & \left[,\ \textbf{"BYT"}\right] \\ \left\{\begin{array}{l}\textbf{\textit{text - variable,}}\\ \textbf{\textit{text - array}}\end{array}\right\} & ,\textbf{"MAX"} \\ \left\{\begin{array}{l}\textbf{\textit{text - variable}}\\ \textbf{\textit{arithmetic - variable}}\\ \textbf{\textit{array}}\end{array}\right\} & ,\textbf{"DIM"} \\ \textbf{\textit{array}},\ \textbf{\textit{dimension}} & \end{array}\right\}\right)$$

### *text-expression*

**Description** *Optional.* MANTIS returns the current length of the expression you specify.

**Format** Text expression.

### “BYT”

**Description** *Optional.* MANTIS returns the number of bytes for a text expression.

**Consideration** This parameter is supported by Releases 5.2 and above of MANTIS for the IBM mainframe.

### *text-variable*, “MAX”
### *text-array*, “MAX”

**Description** *Optional.* MANTIS returns the maximum length (in characters) of the text variable or array element whose name you specify.

**Format** Text variable or text array name.

**Consideration** This parameter is invalid for numeric fields.

***text-variable*, "DIM"**
***arithmetic-variable*, "DIM"**
***array*, "DIM"**

**Description** *Optional.* MANTIS returns the number of dimensions for the text variable, arithmetic variable or array name you specify.

**Format** Text variable, arithmetic variable or array name.

***array,dimension***

**Description** *Optional.* MANTIS returns the value of the specified dimension in the specified array.

**Format** Array name and dimension number.

**Examples**

-
  ```
  SIZE(CUST NAME,"MAX")
  ```
- If OPTION is numeric (BIG or SMALL), enter:

  ```
  SHOW SIZE(OPTION,"DIM"),SIZE(OPTION,1),SIZE(OPTION,2)
  ```

  The above statement will show the number of dimensions and maximum subscript value defined for each subscript.
- If NAME is TEXT, enter:

  ```
  SHOW SIZE(NAME,"DIM"),SIZE(NAME,1),SIZE(NAME,"MAX")
  ```

  The above statement will show the dimensions, the number of rows (first dimension), and maximum length currently defined for a text variable.

**General considerations**

- Using SIZE with one parameter returns the number of characters in the TEXT string.
- You can use the SIZE function in programming mode for dimension dissimilarity debugging. For example, when you receive the message "212 TYPE OR DIMENSIONS IN PROFILE ARE INCONSISTENT WITH AN EXISTING VARIABLE", you can determine the existing dimensions of the variable.

## SLICE

Use the SLICE statement to specify the number of statements in a program slice. With the SLOT value, this determines how many statements MANTIS will execute before stopping with the message: POTENTIAL PROGRAM LOOP ENCOUNTERED.

**SLICE *statements***

### *statements*

**Description** *Required.* Specifies the number of statements in a program slice.

**Format** Arithmetic expression that evaluates to a value in the range 1–32767.

**Consideration** If you have not used the SLOT statement to change the default of one program slice per slot, this parameter specifies the number of statements MANTIS will execute before stopping with the message:

```
303 POTENTIAL PROGRAM LOOP ENCOUNTERED. ENTER 'KILL' TO
  TERMINATE.
```

**Examples**

```
10 SLICE 3000:SLOT 1
10 SLICE 300:SLOT 10
```

Both of the previous examples allow execution of 3000 statements before MANTIS issues the warning message.

## General considerations

- In MANTIS for the IBM mainframe, the parameters of the SLICE statement affect system performance, but in MANTIS for Windows they do not.
- Do not use SLICE as a debugging device for single stepping as you can in MANTIS for the IBM mainframe. Use breakpoints with the GO command instead.
- MANTIS resets the statement count to zero on every terminal input operation.
- The “Potential program loop encountered” condition does not occur when the program is bound. MANTIS only counts program lines executed when executing unbound program lines.
- This statement overrides the default SLICE value set in your User Profile.

## SLOT

Use the SLOT statement to specify the number of program slices MANTIS will execute before stopping with the message: POTENTIAL PROGRAM LOOP ENCOUNTERED. You can then press ENTER to continue running the program or enter KILL to terminate execution.

**SLOT *slices***

### *slices*

**Description**

*Required.* Specifies the number of program slices MANTIS will execute before stopping with the following warning message:

```
303 POTENTIAL PROGRAM LOOP ENCOUNTERED. ENTER 'KILL' TO
  TERMINATE.
```

**Format**

Arithmetic expression that evaluates to a value in the range 1–32767.

**Examples**

```
10 SLICE 3000:SLOT 1
10 SLICE 300:SLOT 10
```

Both of the previous examples allow execution of 3000 statements before MANTIS issues the warning message.

**General considerations**

- This statement is included for compatibility with MANTIS for the IBM mainframe where it is used to influence system performance. It does not affect the performance of a Windows system, so there is no reason for you to change the default value (SLOT 1) specified in your User Profile.
- Do not use SLOT as a debugging device for single stepping as you can in MANTIS for the IBM mainframe.
- This statement overrides the default SLOT value set in your User Profile.

## SMALL

Use the SMALL statement to define numeric variables or arrays of numeric variables, allocating space in the work area to hold their values with up to 6 significant digits. Values are initially set to zero.

**SMALL *small-name*[(*dimension*,...)],...**

### *small-name*

**Description** *Required.* Defines the symbolic name of the numeric variable or array.

**Format** MANTIS symbolic name.

**Consideration** No processing occurs if *small-name* is already defined.

### *dimension*

**Description** *Optional.* Specifies an array dimension. Use one dimension parameter to specify a one-dimensional array, two dimension parameters to specify a two-dimensional array, and so on.

**Format** Arithmetic expression that evaluates to a value in the range 1–16000.

**Consideration** A maximum of 255 dimensions can be specified. Each dimension specifies the maximum value of the corresponding array subscript.

**Example**

```
10 X=15
20 SMALL ALPHA(64,3),BETA(X)
```

**General considerations**

- No processing takes place if *small-name* is already defined.
- You can specify a maximum of 65535 variable names.
- Selected options that reduce the number or size of array dimensions that can be specified, or the number of variable names allowed, can be specified in the Master User library.
- The total size of a single array (including overhead) cannot exceed 64K.

## SQR

Use the SQR (square root) function to return the square root of an arithmetic expression.

**SQR(*a*)**

***a***

**Description** *Required.* Specifies the number whose square root you want returned.

**Format** Valid arithmetic expression that evaluates to a positive number.

**Example**

```
SQR(100)  returns        10
```

## STOP

Use the STOP statement to terminate program execution and return to your facility program unless you are in programming mode.

**STOP**

### Example

```
10 ENTRY DATA_ENTRY
20 .SCREEN MAP1("INDEX")
30 .FILE REC1("INDEX","SERENDIPITY")
40 .CONVERSE MAP1
50 .WHILE MAP1<>"CANCEL"
60 ..DO INSERT_RECORD
70 ..CLEAR MAP1
80 ..CONVERSE MAP1
90 .END
100 .STOP
110 EXIT
120 ENTRY INSERT_RECORD
130 .INSERT REC1
140 EXIT
```

### General considerations

- In programming mode, a STOP in an external routine stops execution within the subroutine. Use the UP command to return to the calling program.
- It is not necessary to put a STOP statement at the end of the main program. MANTIS will terminate execution of the program when it reaches an EXIT statement or when there are no more statements.
- See “External DO” on page 267 of this manual for additional information on the STOP statement and how it works with external DO.

## $SYMBOL

Use the $SYMBOL function to return the text value assigned to the environment variable with a specified name.

**$SYMBOL(*t*)**

***t***

**Description** *Required.* Specifies the name of the environment variable.

**Format** Must be a text expression that evaluates to a valid environment variable name, as discussed in *MANTIS for Windows Administration Guide*, P19-2304.

**Example**

```
SET $SYMBOL "MONTH" TO "NOVEMBER"
SHOW $SYMBOL("MONTH")
NOVEMBER
```

**General consideration**

$SYMBOL is not supported by MANTIS for the IBM mainframe.

## TAN

Use the TAN function to return the tangent of an arithmetic expression in radians.

**TAN(*a*)**

***a***

**Description** *Required.* Specifies the number whose tangent you want returned.

**Format** Valid arithmetic expression.

**Example**

```
100 Z=X**3+Y
.
.
.
200 CORNER=TAN(Z)
```

## TERMINAL

TERMINAL is provided for compatibility with MANTIS for the IBM mainframe.  It returns the computer name of the system on which MANTIS is running, as found in the Windows registry.

**TERMINAL**

### Example

```
SHOW TERMINAL
```

# TERMSIZE

Use the TERMSIZE function to return a text value giving the size of the screen in rows and columns in the format *rr*X*cc*.

**TERMSIZE**

## Examples

- TERMSIZE returns the number of rows and columns on the screen, as shown in the following example:

  ```
  SHOW TERMSIZE
  24X80
  ```

- TERMSIZE(1,2) returns the number of row dimensions on the screen, as shown in the following example:

  ```
  SHOW TERMSIZE(1,2)
  24
  ```

- TERMSIZE(4) returns the number of columns on the screen, as shown in the following example:

  ```
  SHOW TERMSIZE(4)
  80
  ```

## TEXT

Use the TEXT statement to define text variables or arrays of text variables.  The current lengths are initially set to zero.

**TEXT *text-name*[([*dimension*,...,]*length*)],...**

### *text-name*

| | |
|---|---|
| **Description** | *Required.*  Specifies the symbolic name of the text variable or array. |
| **Format** | MANTIS symbolic name. |
| **Consideration** | No processing occurs if *text-name* is already defined. |

### *dimension*

| | |
|---|---|
| **Description** | *Optional.*  Specifies an array dimension.  Use one dimension parameter to specify a one-dimensional array, two dimension parameters for a two-dimensional array, and so on. |
| **Format** | Arithmetic expression that evaluates to a value in the range 1–16000. |
| **Consideration** | A maximum of 255 dimensions can be specified.  Each dimension specifies the maximum value of the corresponding array subscript. |

### *length*

| | |
|---|---|
| **Description** | *Optional.*  Specifies the maximum length of a text variable or array element. |
| **Default** | 16 |
| **Format** | Arithmetic expression that evaluates to a value from 1–32750. |

## Examples

```
20 TEXT ALPHA(64,3),BETA(12)
```

Defines a text array ALPHA with 64 3-character elements and a 12-character text variable BETA.

MANTIS accepts only as many characters in a text variable as you specify in the TEXT statement. For example, if you enter:

```
10 TEXT GAMMA,DELTA(5)
20 GAMMA="123456789ABCDEFGHIJK":DELTA="123456789"
30 SHOW GAMMA,DELTA
```

the screen displays:

```
123456789ABCDEFG          12345
```

## General considerations

- ♦ No processing occurs if *text-name* is already defined.
- ♦ You can specify a maximum of 65,535 variable names.
- ♦ The total size of a single array (including overhead) cannot exceed 64K.
- ♦ Selected options that reduce the number or size of array dimensions that can be specified, the number of variable names allowed, and the maximum length of a text string, can be specified in the Master User Library.

## TIME (function)

Use the TIME function to return a text value containing the current time. The format of the time is determined by a previous time mask specification using the TIME statement.

**TIME**

### Example

```
TIME="HH:MM"
SHOW TIME
11:25
.
.
.
30 .TIME="HH:MM:SS"
40 .CUTOFF="10:45:00"
50 .IF TIME>CUTOFF
60 ..SHIP`_DATE=TOMORROW
70 .ELSE
80 ..SHIP`_DATE=DATE
90 .END
.
.
.
```

## General considerations

- The current time mask setting is inherited on an external DO or CHAIN LEVEL, and is restored upon subprogram EXIT. This inheritance does not occur when a nonprivileged user's program attempts to externally DO or CHAIN LEVEL a privileged user's program.
- A CHAIN (without LEVEL) will reset the time mask to NULL (“”).
- For nonprivileged user's programs, if the time mask has not been previously specified, or has been set to NULL (“”), the TIME function returns the current time in the format specified by the TIME MASK general option in the configuration file.
- For privileged user's programs, if the time mask has not been previously specified, or has been set to NULL (“”), the TIME function returns the current time in HH:MM:SS format.
- See TIME (statement), DATE (function), and DATE (statement).

## TIME (statement)

Use the TIME statement to specify the time mask to be used by the TIME function.

**TIME=*mask-expression***

### *mask-expression*

**Description** *Required.* Specifies the time mask to be used by the TIME function.

**Format** Text expression of 0–32 characters.

**Considerations**

- ♦ If the mask specified is longer than 32 characters, the first 32 characters are used.
- ♦ The following special strings are substituted with the data described when the TIME function is invoked. Any character in the mask that is not part of one of these special strings is treated as punctuation and is not translated.
  - *AM* AM/PM indicator
  - *HH* Hours

    12-hour clock (zero suppressed) if AM is specified

    24-hour clock if AM not specified
  - *MM* Minutes
  - *SS* Seconds
- ♦ Where there are ambiguities on substitution, the substitution takes place in the order of the special strings listed above.
- ♦ There is a special mask escape character (%). Any character following this escape is taken literally and the escape can be used to break up what might otherwise be a valid substitution string.
- ♦ When alphabetic substitution is required (e.g., AM), the case of any special string characters is reproduced on substitution. Case is not important for numeric substitution.

## Example

```
TIME="HH:MM:SS am"
SHOW TIME
 5:45:12 pm

TIME="24 hour time is HH:MM:SS"
SHOW TIME
24 hour time is 17:45:12

TIME="Sa%m said it was HH o’clock AM"
SHOW TIME
Sam said it was  5 o’clock PM
```

## General considerations

- ♦ See TIME (function), DATE (function), and DATE (statement).

## TRAP

Use the TRAP statement to determine whether error conditions in GET, UPDATE, INSERT, DELETE, and CALL processing will terminate program execution or whether MANTIS will provide the program with an error status and allow execution to continue.

$$\textbf{TRAP} \begin{Bmatrix} \textbf{\textit{file - name}} \\ \textbf{\textit{access - name}} \\ \textbf{\textit{interface - name}} \end{Bmatrix} \begin{bmatrix} \underline{\textbf{ON}} \\ \textbf{OFF} \end{bmatrix}$$

### *file-name*

| | |
|---|---|
| **Description** | *Optional.* Specifies the symbolic name of a MANTIS file, as defined in a FILE statement. |
| **Format** | MANTIS symbolic name. |

### *access-name*

| | |
|---|---|
| **Description** | *Optional.* Specifies the symbolic name of an external file, as defined in an ACCESS statement. |
| **Format** | MANTIS symbolic name. |

### *interface-name*

| | |
|---|---|
| **Description** | *Optional.* Specifies the symbolic name of an interface, as defined in an INTERFACE statement. |
| **Format** | MANTIS symbolic name. |

### ON

| | |
|---|---|
| **Description** | *Optional.* Indicates that program execution continues after an error (with an error status). |
| **Consideration** | ON is the default for TRAP. |

## OFF

**Description** *Optional.* Indicates that program execution will terminate (with an error message).

**Examples**

♦
```
100 FILE REC("EXAMPLES:CUSTOMERS","CASINO")
.
.
.
160 TRAP REC ON
170 INSERT REC
.
.
.
220 UPDATE REC
230 IF REC="ERROR"
240 .MSG="UPDATE FAILED"
250 END
.
.
.
```

♦
```
20 INTERFACE ITF("INTERFACE1","ALIBABA")
30 TRAP ITF ON
40 CALL ITF
50 IF ITF<>""
60.SHOW ITF,FSI(ITF)
70 END
```

## General considerations

- TRAP...ON enables MANTIS to return a status to the program after execution of a GET, UPDATE, INSERT, or DELETE statement.  The range of statuses is listed below.  These statuses do not necessarily apply to all types of I/O activity, that is, MANTIS file or external file.

| | |
|---|---|
| DATA | A numeric field contains nonnumeric data. |
| DUPLICATE | A record with the same key already exists. |
| ERROR | A physical error occurred, the file is full, the record does not exist, or (for sequential variable-length record files) the record length has changed. |
| LOCK | The password specified in the FILE or ACCESS statement is not valid for the type of access attempted. |

- TRAP *interface-name* ON enables MANTIS to return a status to the program after execution of a CALL statement.  The statuses are:

| | |
|---|---|
| ERROR | Error occurred during the CALL statement. |
| TIMEOUT | CALL was not completed within the time specified in the interface profile. |

  When the status is equal to ERROR, you can get detailed information about the error using the FSI function with the *interface-name*.

- If you do not include a TRAP statement in your program, any of the above error conditions will terminate execution (as if TRAP OFF had been executed).

- If TRAP ON is specified for an external file, the first GET, UPDATE, INSERT, or DELETE statement will also trap errors associated with opening the external file.

## TRUE

Use the TRUE function to return the value TRUE (1).

**TRUE**

### Example

```
100 ..IF MONTH>12
110 ...ERROR=TRUE
120 ..END
```

## TXT

Use the TXT function to return the text value of an arithmetic expression in standard numeric display format.

**TXT(*a*)**

***a***

**Description** *Required.* Specifies the number whose text value you want returned.

**Format** Valid arithmetic expression.

**Example**

```
SHOW TXT(100)
100
```

**General consideration**

TXT(0) returns a single space. For example, SIZE(TXT(0)) is equal to 1.

## UNPAD

Use the UNPAD statement to remove padding characters from a text variable, array element, or substring.

$$\textbf{UNPAD } \textbf{\textit{string}} \left[\textbf{\textit{padding}}\right] \begin{bmatrix}\textbf{BEFORE}\\ \underline{\textbf{AFTER}}\\ \textbf{ALL}\end{bmatrix}$$

### *string*

| | |
|---|---|
| **Description** | *Required.* Specifies the text variable, array element, or substring from which you want to remove padding characters. |
| **Format** | Name of a TEXT variable, the subscripted name of a TEXT array, or a reference to a TEXT substring. |

### *padding*

| | |
|---|---|
| **Description** | *Optional.* Specifies the padding character. |
| **Default** | (blank) |
| **Format** | Text expression. |
| **Consideration** | The first character is the padding character. Any subsequent characters are ignored. |

### BEFORE

| | |
|---|---|
| **Description** | *Optional.* MANTIS removes padding characters from the beginning of the string. |

### AFTER

| | |
|---|---|
| **Description** | *Optional.* Removes padding characters from the end of the string. |

## ALL

**Description** *Optional.* Removes padding characters from both ends of the string.

**Consideration** If you do not specify BEFORE, AFTER, or ALL, MANTIS removes padding characters from the end of the string.

**Examples**

- Strip leading and/or trailing spaces:

```
UNPAD CLIENT NAME AFTER    "JOE JACKSON          "    =  "JOE
  JACKSON"
UNPAD CLIENT NAME BEFORE   "          JOE JACKSON"    =  "JOE
  JACKSON"
UNPAD CLIENT NAME ALL      "      JOE JACKSON    "    =  "JOE
  JACKSON"
```

- Determine if a text field is all blanks or null:

```
UNPAD CLIENT NAME
IF CLIENT NAME=""
.
.
.
END
```

- Determine if a text field is all zeros or null:

```
UNPAD CLIENT NUMBER "0"
IF CLIENT NUMBER=""
.
.
.
END
```

## General considerations

- When the string is not a substring, the UNPAD statement removes all consecutive characters that match the padding character from the specified end(s) of the string. The current length of the string is reduced accordingly.
- You cannot use BEFORE, AFTER, or ALL when removing padding from a substring.
- You can remove padding from a substring by specifying one or two subscripts (see "Text substrings" on page 51).
- If the first and last character positions are specified, consecutive padding characters are removed from the end of the substring, and any characters to the right of the substring are moved left to close the gap.
- The current length of the string is reduced accordingly.

## UNTIL-END

Use the UNTIL-END statements to execute a block of statements repeatedly until a specified condition becomes true.  MANTIS executes the block of statements in the range of the UNTIL-END once before it tests the condition.

**UNTIL *expression***

**.**

**statements**

**.**

**END**

### expression

**Description**

*Required.*  Specifies the condition that will end execution of the UNTIL loop when TRUE.

**Format**

Valid arithmetic expression that evaluates to TRUE (nonzero value) or FALSE (zero value).

**Example**

```
10 UNTIL YEAR=2000
20 .SHOW "ENTER YEAR IN FORMAT YYYY"
30 .OBTAIN YEAR
40 END
```

**General consideration**

UNTIL-END always executes the enclosed statements at least once.

# UPDATE

Use the UPDATE statement to update the contents of a record in a MANTIS file or in an external file. You need not read a record from a file before updating it.

## MANTIS file UPDATE

**UPDATE *file-name* [LEVEL=*level-number*]**

### *file-name*

**Description** *Required.* Specifies the symbolic name of the file you want to update.

**Format** MANTIS symbolic name defined in a previously executed FILE statement.

### LEVEL=*level-number*

**Description** *Optional.* Specifies which level of MANTIS array elements contains the updated values of the record contents.

**Default** LEVEL=1

**Format** Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding FILE statement.

**Example**

```
60 ..CONVERSE MAP
70 ..WHEN MAP="PF1"
80 ...INSERT RECORD
90 ..WHEN MAP="PF2"
100 ...DELETE RECORD
110 ..WHEN MAP="PF3"
120 ...UPDATE RECORD
130 ..END
140 ..GET RECORD
150 .END
160 EXIT
```

## General considerations

- You do not need to read a record from a MANTIS file before updating it.
- MANTIS obtains the value of each field in the file from the corresponding MANTIS variable or array element and writes it to the record indicated by the key fields.
- The status of the UPDATE can be obtained by using the symbolic name *file-name* to invoke a built-in text function. The status is a null text value if the UPDATE was successful, or HELD if MANTIS could not update the record because another user has locked the record with the GET ENQUEUE statement. If the UPDATE is unsuccessful, you can use the HELP LAST command (in Program Design) to examine the error message that may have been returned. This information is also available to MANTIS fault handling routines (see “SET” on page 202). MANTIS may also return a status of LOCK or ERROR if TRAP is in effect.

| | |
|---|---|
| LOCK | The password specified in the FILE statement for this file view is not valid for updates. |
| ERROR | A physical error occurred while updating the record; or the record to be updated did not exist on the file. |

- Do not change a key field in a record before an UPDATE; instead, delete the existing record and insert the new record with an altered key.
- MANTIS performs an implicit COMMIT on every terminal input operation, unless this functionality is disabled by the COMMIT OFF statement.

## External file UPDATE

**UPDATE *access-name* [LEVEL=*level-number*]**

### *access-name*

**Description** *Required.* Specifies the symbolic name of the external file you want to update.

**Format** MANTIS symbolic name defined in a previously executed ACCESS statement.

### LEVEL=*level-number*

**Description** *Optional.* Specifies which level of MANTIS array elements contains the updated values of the record contents.

**Default** LEVEL=1

**Format** Arithmetic expression that evaluates to a value in the range 1–*levels*, where *levels* is the number of levels specified in the corresponding ACCESS statement.

**Example**

```
 20 .ACCESS RECORD("INDEX","SERENDIPITY",16)
 30 .SCREEN MAP("INDEX")
 40 .CONVERSE MAP
 50 .COUNTER=1
 60 .WHILE MAP<>"CANCEL" AND COUNTER<17
 70 ..WHEN INDICATOR(COUNTER)="G"
 80 ...GET RECORD LEVEL=COUNTER
 90 ..WHEN INDICATOR(COUNTER)="U"
100 ...UPDATE RECORD LEVEL=COUNTER
.
.
```

**General considerations**

- MANTIS obtains the value of each field in the external file view from the corresponding MANTIS variable or array element and writes it to the record to be updated.
- You do not need to read the record before you update it, but before you issue the UPDATE statement, make sure that the MANTIS variables for each field in the external file view contain the right values.

- For INDEXED files, the contents of key data elements identify the record to be updated.
- For SEQUENTIAL files, the associated reference variable (which must be defined during External File View Design) contains the Relative Byte Address (RBA) that identifies the record to be updated. You cannot update a variable-length record if you have changed its length.
- For NUMBERED files, the associated reference variable (defined during External File View Design) contains the Relative Record Number (RRN) that identifies the record to be updated.
- Do not change a key field in a record in an INDEXED file; instead, delete the existing record and insert the new record with an altered key.
- The status of the UPDATE can be obtained by using the symbolic name *access-name* to invoke a built-in text function. The status is a null text value if the UPDATE was successful, or HELD if MANTIS could not update the record because another user has locked the record with the GET ENQUEUE statement. If the UPDATE is unsuccessful, you can use the HELP LAST command (in Program Design) to examine the error message that may have been returned. This information is also available to MANTIS fault handling routines (see "SET" on page 202). MANTIS may also return a status of LOCK or ERROR if TRAP is in effect.

| | |
|---|---|
| LOCK | The password specified in the ACCESS statement for this file view is not valid for updates. |
| ERROR | A physical error occurred while updating the record, or the record to be updated did not exist on the file. |

- MANTIS performs an implicit COMMIT on every terminal input operation, unless this functionality is disabled by the COMMIT OFF statement.
- When updating a record into a file with multiple keys, a duplicate on any key will cause the UPDATE to fail.
- MANTIS performs the update operation by deleting the original record and inserting a new record. Therefore, any failure on the insert means that the record no longer exists, for example, a duplicate of an alternate key.

## UPPERCASE

Use the UPPERCASE function to return the text value of a text expression after changing any lowercase letters to uppercase.

**UPPERCASE(*t*)**

***t***

**Description** *Required.* Specifies the text value to be converted to uppercase.

**Format** Valid text expression.

**Example**

```
SHOW  UPPERCASE("Lend Me Your Ears")
LEND ME YOUR EARS
```

**General consideration**

The UPPERCASE function is not supported by MANTIS for the IBM mainframe.

## USER

Use the USER function to return a text value containing the user name entered during MANTIS sign-on.

**USER**

**Format** 1–16 characters.

**Example**

```
 10  ENTRY LIST_ENTRIES
 20
 .
 .
 .
 80  .FILE REC(ENTITY,PASSWORD,4)
 90  .HEAD"ENTITY"+ENTITY+"FOR_USER"+USER
100  .GET REC FIRST LEVEL=1
```

## USERWORDS

Use the USERWORDS function to return the number of MANTIS symbolic names currently in use.

**USERWORDS**

### Example

```
SHOW USERWORDS
```

## VALUE

Use the VALUE function to return the numeric value of a text expression.

**VALUE(*t*)**

***t***

**Description** *Required.* Specifies a text expression for which you want a numeric value returned.

**Format** Valid text expression.

**Example**

```
.
.
.
40 .DO BROWSE_RTN
50 .CONVERSE MAP
60 .WHILE MAP<>"CANCEL"
70 ..WHEN VALUE(CMD(1,5))>ZERO AND VALUE(CMD(1,5))<99999
80 ...CODE=VALUE(CMD(1,5))
90 ...DO REPOINT(MAP,REC,CODE)
.
.
.
```

**General consideration**

MANTIS ignores all nonnumeric characters except for decimal point (.), plus sign (+), minus sign (-), and exponent indicator (E).

## WAIT

Use the WAIT statement to suspend execution of your program until you press ENTER or any other action key. Use a WAIT statement to prevent data displayed by SHOW from being overwritten by a subsequent statement before you have had time to read it.

**WAIT**

### Example

```
90 ENTRY UPDATE RECORDS
91 .SHOW "YOU ARE NOW IN UPDATE RECORDS ROUTINE"
92 .SHOW "PRESS ENTER TO CONTINUE"
93 .WAIT
94 EXIT
```

### General considerations

- ♦ PROMPT and CONVERSE completely overwrite the screen as does STOP if it returns to a selection menu. Executing a WAIT before these statements delays their execution until you indicate (by pressing ENTER or any action key) that you have read any data previously displayed by SHOW.
- ♦ If you execute an OBTAIN statement to obtain input data between SHOW and PROMPT, CONVERSE or STOP, a WAIT statement is unnecessary.

## WHEN-END

Use the WHEN-END statements to execute a block of statements when a specified condition is met. MANTIS performs the test before executing the block of statements. If the condition is FALSE, execution drops through to the next WHEN or to the END statement.

---

**WHEN *expression***

**.**

**statements**

**.**

**WHEN *expression***

**.**

**statements**

**.**

**END**

---

### *expression*

**Description** *Required.* Specifies the condition which must be true for MANTIS to execute the following block of statements.

**Format** Arithmetic expression that evaluates to TRUE (nonzero value) or FALSE (zero value).

## Example

```
 10 ENTRY INDEX
 20 .FILE RECORD("INDEX","SERENDIPITY")
 30 .SCREEN MAP("INDEX")
 40 .GET RECORD
 50 .WHILE RECORD<>"END" AND MAP<>"CANCEL"
 60 ..CONVERSE MAP
 70 ..WHEN MAP="PF1"
 80 ...INSERT RECORD
 90 ..WHEN MAP="PF2"
100 ...DELETE RECORD
110 ..WHEN MAP="PF3"
120 ...UPDATE RECORD
130 ..END
140 ..GET RECORD
150 .END
160 EXIT
```

## General consideration

After a TRUE condition, execution continues with the next WHEN statement.  MANTIS evaluates every WHEN statement unless a BREAK statement is used.

## WHILE-END

Use the WHILE-END statements to execute a block of statements repeatedly while a specified condition is TRUE.  If the condition is FALSE, MANTIS terminates the WHILE and executes the statement after END. If the condition is TRUE, MANTIS executes the statements within the WHILE range and then reevaluates the condition.

**WHILE *expression***

**.**

**statements**

**.**

**END**

### *expression*

| | |
|---|---|
| **Description** | *Required.*  Specifies the condition that must be TRUE while MANTIS executes the block of statements. |
| **Format** | Arithmetic expression that evaluates to TRUE (nonzero value) or FALSE (zero value). |

## Example

```
 10 ENTRY INDEX
 20 .FILE RECORD("INDEX","SERENDIPITY")
 30 .SCREEN MAP("INDEX")
 40 .GET RECORD
 50 .WHILE RECORD<>"END" AND MAP<>"CANCEL"
 60 ..CONVERSE MAP
 70 ..WHEN MAP="PF1"
 80 ...INSERT RECORD
 90 ..WHEN MAP="PF2"
100 ...DELETE RECORD
110 ..WHEN MAP="PF3"
120 ...UPDATE RECORD
130 ..END
140 ..GET RECORD
150 .END
160 EXIT
```

## ZERO

Use the ZERO function to return the value zero.

**ZERO**

### Example

```
10 ENTRY NEW_EXAMPLE
20 .FILE RECORD("JACKSON","CASINO",PREFIX)
30 .TEXT STATE(8),COUNTY(12),TOWN(12)
40 .HEAD "POPULATION REPORT":SHOW " "
50 .CLEAR
60 .GET RECORD
70 .NATION COUNTER=ZERO
.
.
.
```

# 4

# Programming techniques

This chapter offers suggestions for using the MANTIS logical terminal interface, external DO, and other features offered by MANTIS.

## MANTIS Logical Terminal Interface

MANTIS performs terminal I/O through a logical terminal interface that supports a logical display area of 32767 rows by 32767 columns. This display area enables you to design and converse screens (maps) that are larger than your physical screen. A single screen design can be as large as 255 rows by 254 columns or 254 rows by 255 columns. Your physical screen acts as a movable window on the logical display, and you can scroll around the logical display by using the keys described in the table in "MANTIS editing and windowing keys" on page 32.

NOTE

Windowing in MANTIS for Windows is relative to the size of the current map set (you can scroll only to the edge of the map set). Windowing in MANTIS for the IBM mainframe is restricted to the size of the logical display (you can scroll to anywhere in the logical display area, regardless of the map set size.)

The MANTIS Screen Design Facility enables you to design and save screens for use in your application. The MANTIS Program Design Facility enables you to build programs that send your screen designs to physical devices (such as a physical screen and printer).

## Building a map set in your program

In your applications, you can add individual maps to the logical display to form a map set. Maps within a map set are displayed according to their entry sequence into the map set as specified in programming statements. Maps within a map set overlay each other; the top map is known as the active map. Every field on this map that falls within the current window is displayed. All other maps in the map set are passive maps.

A map's position within the map set is controlled by the first set of parameters on the CONVERSE statement:

$$\textbf{CONVERSE}\ \textbf{\textit{screen - name}}\left[\begin{array}{l} \textbf{[(}\textbf{\textit{row1,col1}}\textbf{)]}\left[\begin{array}{l}\textbf{WAIT}\\ \textbf{SET}\\ \textbf{UPDATE}\end{array}\right]\left[\left\{\begin{array}{l}\textbf{WINDOW}\\ \textbf{DISPLAY}\end{array}\right\}\right]\\ \textbf{[(}\textbf{\textit{row2,col2}}\textbf{)] [template]}\\ \textbf{RELEASE} \end{array}\right]$$

The WAIT, SET, and UPDATE options on the CONVERSE statement determine whether a terminal I/O occurs (whether the map is displayed), and whether the displayed input fields on the passive maps are protected or unprotected. A CONVERSE statement overrides an ATTRIBUTE(UNPROTECTED) statement in your program, which overrides a protected field attribute specified in Screen Design.

Maps can be conversed over any part of the logical display, including over headings or data fields. You can create error message maps or a single map into which you write different text in different situations.

Because maps are translucent by default, all fields not completely overlaid by the other maps will show through the active map. When conversing maps using the UPDATE option of the CONVERSE statement, the following governs whether you can update partially displayed fields on a passive map:

- Fields on a passive map that are partially displayed because they are overlaid by another map, cannot be updated; fields on a passive map that are partially displayed because they overlap the physical screen boundary, can be updated as long as the AUTOMATIC WINDOWING attribute is specified for the active map.
- In MANTIS for the IBM mainframe, they can always be updated with a CONVERSE UPDATE. AUTOMATIC WINDOWING should be set to Y (yes) for compatibility with MANTIS for the IBM mainframe (refer to the *MANTIS for Windows Facilities Reference Manual*, P19-2301).

The following list describes how the CONVERSE statement uses the WAIT, SET, and UPDATE options:

- **CONVERSE *screen-name*** without a WAIT, SET, or UPDATE creates a new map set. This map set contains only the named screen and causes a terminal I/O to occur, thereby displaying the map on your screen. The row/column coordinates (the default is row 1, column 1) specify the row and column positions for the screen.

- **CONVERSE *screen-name* WAIT** adds the named map to the current map set but does not cause a terminal I/O to occur. The named map is not displayed.

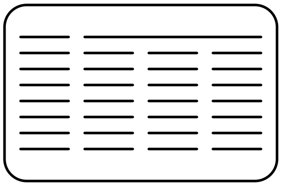
- **CONVERSE *screen-name* SET** adds the named map to the current map set and causes a terminal I/O to occur. This is the active map in the map set (it will appear on top of the other maps). This means overlapping fields from the active map have display precedence.

- **CONVERSE *screen-name* UPDATE** acts the same as CONVERSE *screen-name* SET, except all data entry fields that are fully displayed from underlying maps are unprotected. Unprotected fields that extend beyond the boundaries of the physical screen do not need to be completely displayed to be updated, provided the AUTOMATIC WINDOWING attribute is specified for the active map.

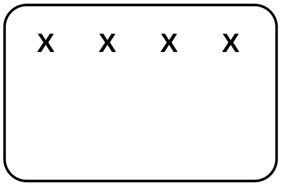
The final map display can consist of a single map or a composite set of maps positioned dynamically within the logical display. If the map display is larger than the physical screen, the operator can treat the physical screen as a window, moving it around the logical display by using windowing keys (see the table in “MANTIS editing and windowing keys” on page 32).

## CONVERSE

Using the following screen designs, see how one screen overlays the other:

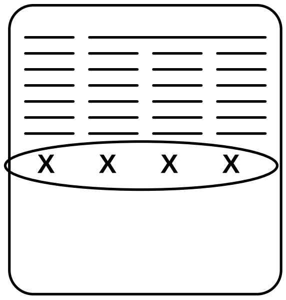


**Report Screen**



**Division Subtotals**

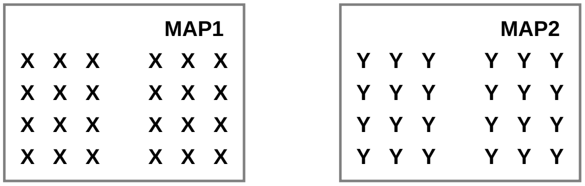
When you converse the maps into one display, you obtain the following results:



**Report Screen**

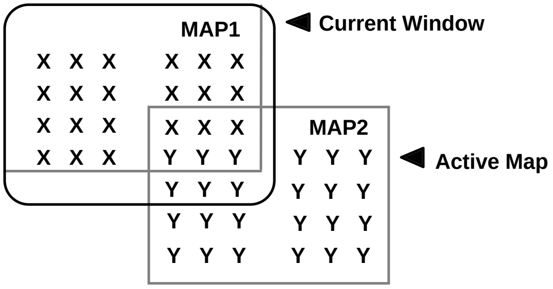
**Division Subtotals**

Using the following maps and program, more options of the CONVERSE statement are demonstrated, looking at active and passive maps:



```
 10 ENTRY CLIENT_ENTRY
 20 .SCREEN MAP1("NEW_CLIENT")
 30 .SCREEN MAP2("PAGE_2")
.
 70 .CONVERSE MAP1
.
120 ...IF FIELD="Y"
160 ....DO EXTRA_INFO
.
300 ENTRY EXTRA_INFO
310 .UNTIL MAP2="CANCEL" OR MESSAGE=""
320 ..CONVERSE MAP2(10,45) SET
.
400 .IF MAP2="ENTER"
410 ..INSERT REC
420 ..CLEAR
```

The CONVERSE statement in line 320 produces the following results:

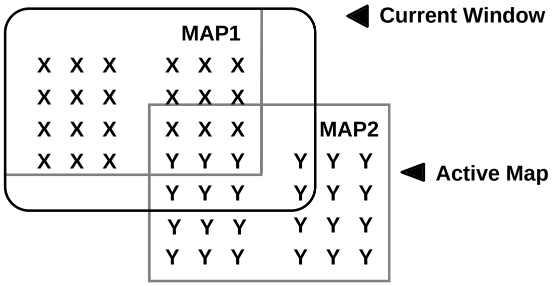


NOTE

The fully displayed fields on MAP1 are protected because of the SET parameter used in the CONVERSE statement.  Also, the last field on MAP1 is overlaid by the first field on MAP2.

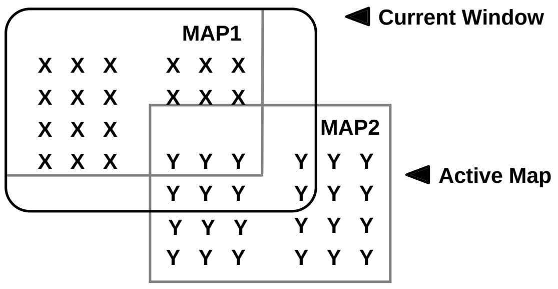
The following illustration shows a variation on the CONVERSE statement in line 320. This variation uses the UPDATE option instead of the SET option.

```
CONVERSE MAP2(10,45) UPDATE
```

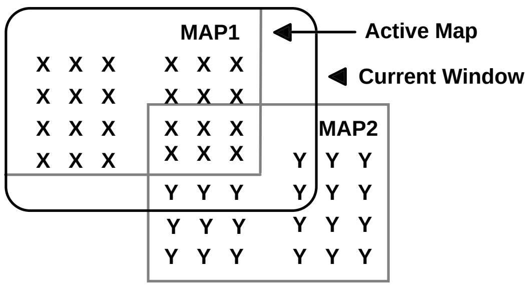


The fully displayed input fields on MAP1 are unprotected.

You can create an opaque map by using the Opaque option in the Library Functions of Screen Design. The active opaque map will completely overlay the parts of the passive map it covers.

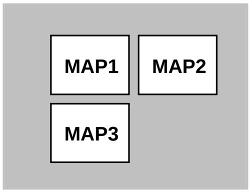


Executing a CONVERSE MAP1 UPDATE again produces the following result:

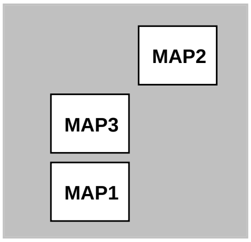


## Multiple images of a single-screen design

You can converse a screen only once in a map set. If you converse a screen a second time and specify row and column coordinates different from those in the first CONVERSE statement, the screen will move to the new location within the map set. In the following example, the CONVERSE statement in line 210 moves MAP1 from its original position (1,1) to its new position (60,1):



```
110  CLEAR
120  CONVERSE MAP2 (1,80)
130  CONVERSE MAP1 WAIT
140  CONVERSE MAP3 (30,1) SET
  .
  .
  .
```



```
210  CONVERSE MAP1(60,1) WAIT
```

To make one screen to appear multiple times within one logical display, define a new SCREEN variable for it each time you want it to appear. In the following example, MAP1 and MAP2 are identical:

```
   10 SCREEN MAP1("NAME"), MAP2("NAME")
   .
   .
   .
  100 .CONVERSE MAP1 WAIT
  110 .CONVERSE MAP2 (45,1) WAIT
  120 .CONVERSE MAP3 (15,1) SET WINDOW (15,1)
```

Each field on a screen has a unique name and value. If you are using one screen design more than once in your logical display, you will probably want to assign multiple values to a field that appears in each screen occurrence. To do so, you must prefix each successive occurrence of the screen, as follows:

```
  100 SCREEN MAP1("NAME",PREFIX)
```

In the previous example, MAP1 is the prefix that must be changed for each occurrence of the screen.

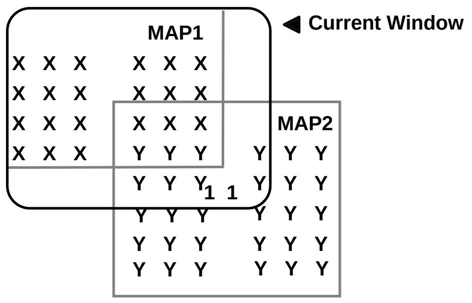
## Windowing

Now, we will extend the CONVERSE statement to determine the physical display position of the map set.  Because the logical display is 32676 columns wide and 32767 rows long, and you can design a screen with as many as 255 rows or 255 columns, your map or map set may overrun the physical screen boundaries.  By using the last parameters of the CONVERSE statement, you can specify whether the row and column coordinates will be displayed and where the window will be located on the map set:

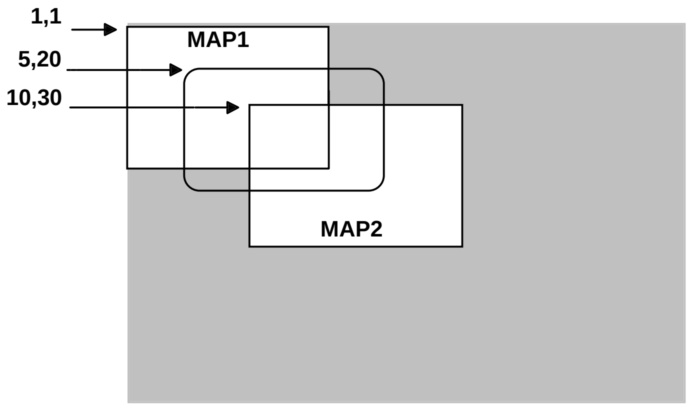
$$\textbf{CONVERSE } \textbf{\textit{screen - name}} \left[ \begin{array}{l} \textbf{[(}\textbf{\textit{row1, col1}}\textbf{)]} \left[ \begin{array}{l} \textbf{WAIT} \\ \textbf{SET} \\ \textbf{UPDATE} \end{array} \right] \left[ \left\{ \begin{array}{l} \textbf{WINDOW} \\ \textbf{DISPLAY} \end{array} \right\} \right] \\ \textbf{[(}\textbf{\textit{row2, col2}}\textbf{)] [template]} \\ \textbf{RELEASE} \end{array} \right]$$

CONVERSE *screen-name* WINDOW(*row*,*col*) displays the map with the upper corner of the window positioned at the specified row and column coordinates of the logical display. These coordinates are displayed at the bottom of the screen, and can optionally be displayed by pressing WINDOWMAP. Sample displays are shown below. Note the fully displayed input fields on MAP1 are protected with the SET option and unprotected with the UPDATE option:

```
CONVERSE MAP1 WAIT
CONVERSE MAP2(10,45) SET WINDOW
CONVERSE MAP2(10,45) UPDATE WINDOW
```



```
CONVERSE MAP2(10,30) SET DISPLAY (5,20)
```



CONVERSE *screen-name* DISPLAY(*row*,*col*) enables you to position the window at the supplied location and does not display the window coordinates.

All windowing occurs while the MANTIS program executes a single CONVERSE statement.  You can reposition the window by overtyping the row and column coordinates with the following:

- New values and pressing ENTER.  For example, enter 20,40 to scroll to row 20, column 40.
- Displacement values (*+* or - in the first character position and the displacement value in the following character positions).  For example, enter +10 -20 to scroll down 10 rows and left 20 columns.
- Incremental values (i in the first position and the incremental value in the following position).  For example, enter i20 i80 to scroll down 20 rows (WINDOWN) or up 20 rows (WINUP) and right 80 columns (WINRIGHT) or left 80 columns (WINLEFT).  You can modify window scrolling amounts within your program by using the SCROLL statement (e.g., SCROLL 30,30).

The application program cannot restrict the positioning of the window by the operator.  So, the operator can override the key scrolling amounts specified in the program.  The program can choose meaningful scrolling amounts for the initial scroll values.

The application program can also use the DISPLAY (*row, column*) parameters to position the window somewhere other than row 1, column 1 (upper left corner) of the logical display.  DISPLAY does not display the row and column window coordinates.

When you move the window, your screen is updated to show the contents of the logical display at the new window position.

You may notice, however, that some parts of your screen remain unchanged. For compatibility with MANTIS for the IBM mainframe, MANTIS for Windows uses a separator to simulate the invisible attribute byte that precedes every field displayed on a mainframe terminal. The position in front of every field on a map is reserved, by default, for the field separator.

When the field separator of a visible field (at the left of the screen) falls outside the window, MANTIS uses column one for the invisible field separator. Consequently, column one of the screen is always blank for maps designed with field separators. For maps designed without the field separator option, fields on a map are visible in column one.

## Clearing a map

A conversed map remains in the map set until the map set is cleared with a CLEAR statement or with a CONVERSE without a WAIT, SET, or UPDATE option. A CLEAR statement without a map name clears all conversed maps from the map set and sets the KEY function to CLEAR. CLEAR with a map name clears data from the named map's data fields and sets the key value of the named map to null. It does not remove the named map from your map set.

CLEAR or CONVERSE (without a WAIT, SET, or UPDATE) clears the logical display of all existing maps. If you do not want to keep track of the first map set building CONVERSE statement, use a CLEAR statement prior to any CONVERSE. For example:

```
100 WHILE KEY<>"CANCEL"
110 .CLEAR
120 .WHEN NAME="Y"
130 ..CONVERSE MAP_NAME WAIT
140 .WHEN OPTION="Y"
150 ..CONVERSE MAP_OPTION WAIT
160 .WHEN REFERENCE="Y"
170 ..CONVERSE MAP_REFERENCE WAIT
180 .END
190 .CONVERSE MAP_SELECT SET
200 END
```

The CONVERSE statements (lines 130, 150, and 170) require no considerations if they are the first statements executed. CLEAR (line 110) clears any previous map set, so these conversed maps do not overlay an existing map. The CONVERSE statement in line 190 sends the map set to the logical display.

## Clearing a map set

A map set built within your program remains in the logical display until you clear the logical display with a CLEAR statement, or issue a CONVERSE RELEASE statement for any maps that are in the map set. A map defined in an externally done program and not released before returning to the originating program causes the entire map set to be cleared.  In the following example, the fields in MAP1 can be updated:

```
ENTRY PROG1
.
.
.
CONVERSE MAP1 WAIT
DO PROG2
.
.
.
ENTRY PROG2
SCREEN MAP2
CONVERSE MAP2 UPDATE
```

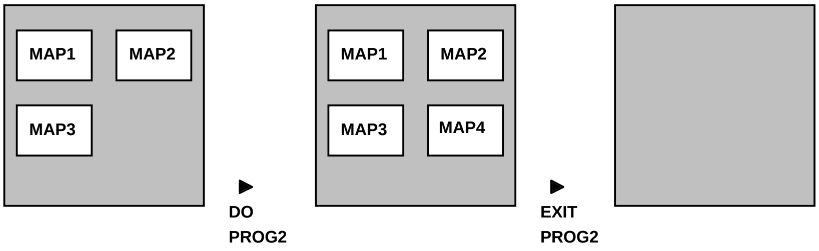
The following examples show how map names are passed to the external program by naming them as arguments of the originating program’s DO statement and as parameters of the external program’s ENTRY statement.  The DO statement in PROG1 passes MAP2 and MAP3 as arguments to PROG2.  PROG2 also uses MAP4, which is not defined in PROG1.

In the following example, MAP4 is not cleared before exiting PROG2. Because MAP4 remains in the map set and is not defined in PROG1, the entire map set is cleared when returning to PROG1.

```
ENTRY PROG1
. SCREEN MAP1("ONE")
. SCREEN MAP2("TWO")
. SCREEN MAP3("THREE")
. CONVERSE MAP1 WAIT
. CONVERSE MAP2(1,20) WAIT
. CONVERSE MAP3(20,1) SET
. DO PROG2(MAP2,MAP3)
```

```
ENTRY PROG2(MAP2, MAP3)
. SCREEN MAP4("FOUR")
. CONVERSE MAP2 (1,20) WAIT
. CONVERSE MAP3 (20,1) WAIT
. CONVERSE MAP4(20,20) SET
EXIT
```

***(reenter PROG1 at statement after DO)***

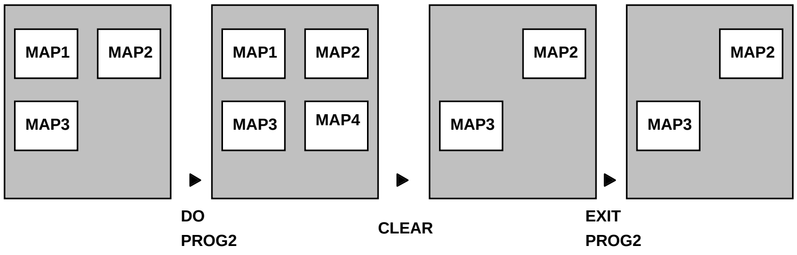


In the next example, PROG2 converses MAP4 and then clears the entire map set. After reconversing MAP2 and MAP3, they are the only maps left in the map set when returning to PROG1.

```
ENTRY PROG1
. SCREEN MAP1("ONE")
. SCREEN MAP2("TWO")
. SCREEN MAP3("THREE")
. CONVERSE MAP1 WAIT
. CONVERSE MAP2(1,20) WAIT
. CONVERSE MAP3(20,1) SET
. DO PROG2(MAP2,MAP3)
```

```
ENTRY PROG2(MAP2, MAP3)
. SCREEN MAP4("FOUR")
. CONVERSE MAP2 (1,20) WAIT
. CONVERSE MAP3 (20,1) WAIT
. CONVERSE MAP4(20,20) SET
. CLEAR
. CONVERSE MAP2 (1,20) WAIT
. CONVERSE MAP3 (20,1) SET
EXIT
```

***(reenter PROG1 at statement after DO)***



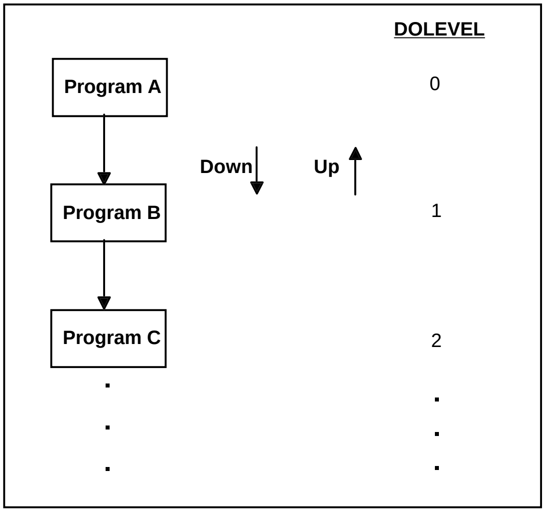
## External DO

Any program can use both internal and external DO statements. Internal routines are defined on ENTRY statements within the original program. Internal DO enables a MANTIS program to transfer data and control of execution t a routine within the same program and return to the next statement following the DO. External DO works similarly to internal DO except that the subroutine resides externally to the program containing the DO statement. External routines are defined in PROGRAM statements in the original program.

External DO can improve the efficiency of execution and make it easier to maintain your applications by enabling many programs to share common subroutines. External DO enables other users to share your programs and access both internal and external subroutines. You can also link subroutines with external DO. The following reserved words support the external DO:

- **PROGRAM**. Identifies the external MANTIS programs to be invoked.
- **DOLEVEL.** Returns a value identifying which level an external program is currently executing. The original program has a DOLEVEL of zero. With each nested external DO statement, the DOLEVEL value increases by one.
- **DOWN and UP**. Enable to list and edit an external subroutine in the work area. DOWN selects an external subroutine at a lower level. UP selects an external subroutine at a higher level.

The following figure shows DOLEVELs in relation to the DOWN and UP commands:



## Passing arguments

When MANTIS passes arguments, it passes variables by reference only, not by copying the actual data.  Variable names for entities can also be passed.  Before using a symbolic name in a DO statement, you must define it using a BIG, SMALL, TEXT, LET, SCREEN, FILE, ACCESS, INTERFACE, SHOW, or OBTAIN statement.  A maximum of 255 arguments can be passed in one DO statement, and only passed arguments can be referenced outside the routine where they are defined.  Individual fields within SCREEN, FILE, ACCESS, and INTERFACE variables must be passed as separate arguments.

Data referenced by parameters are global to both the original program and the subroutine.  Modified values are retained on return to the original program.  Argument passing can affect automatic mapping.  See “Automatic mapping” on page 276 for details.

The names of any variables passed in the originating program's DO statement need not match the names of the parameters in the external program's ENTRY statement.  The following ENTRY statement accepts MAP1, MAP2, FIELD1, and FIELD2 as M1, M2, F1, and F2:

```
DO PROG2(M1,F1,M2,F2)
.
.
.
ENTRY PROG2(M1,F1,M2,F2)
```

Once you pass the maps, you can converse or clear them and update variables within the external program (PROG2).  The passed maps and variables need not be reinitialized with SCREEN or LET commands.

In order to enter variables into previously existing maps while executing an external DO, you must perform another CONVERSE.  To enter data in multiple maps, CONVERSE UPDATE the last conversed map, or CONVERSE UPDATE a map defined within the program executed by an external DO over the existing maps.

## Program architecture

The PROGRAM statement defines the PROGRAM variable, that contains the library name and password supplied.

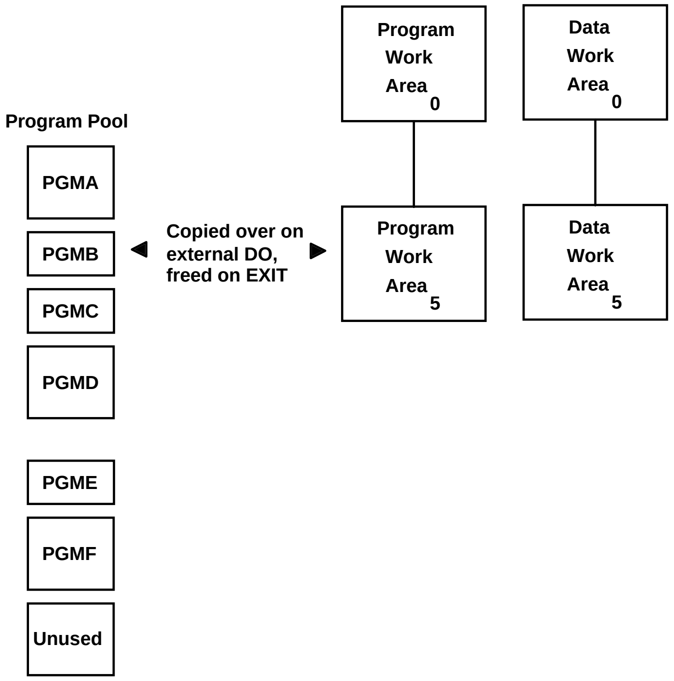
When MANTIS encounters a DO statement, it loads the program and allocates a new program work area and data work area.  MANTIS begins executing the program from its ENTRY statement.

When the external routine encounters EXIT, its program and data work areas are released.  Processing resumes with the statement following the DO in the previous level's program work area.

The program pool optionally holds the code for all programs that have been loaded by execution of an external DO statement.  The program pool is only released when you issue a QUIT, NEW, or CHAIN in programming mode, or STOP, KILL, or CHAIN on a program running outside programming mode, or RELEASE.

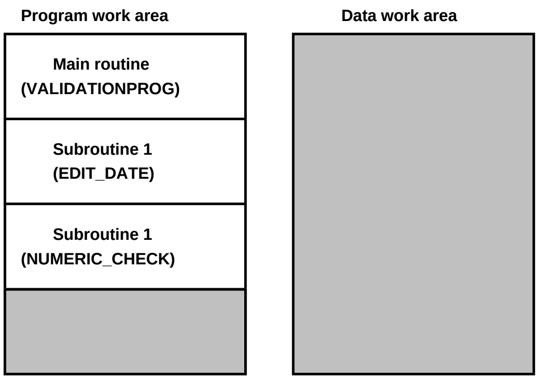
External DO maintains context for the task, along with all associated program storage, in main memory for as long as possible.  In this way, external DO reduces disk access to the MANTIS cluster.

The following figure illustrates the activity of a program that contains an external DO:



## Internal DO vs. external DO vs. CHAIN

Understanding the characteristics of external DO, internal DO, and CHAIN will help you apply the programming guidelines in "Miscellaneous programming hints" on page 281. The following figure presents an example of an internal DO:

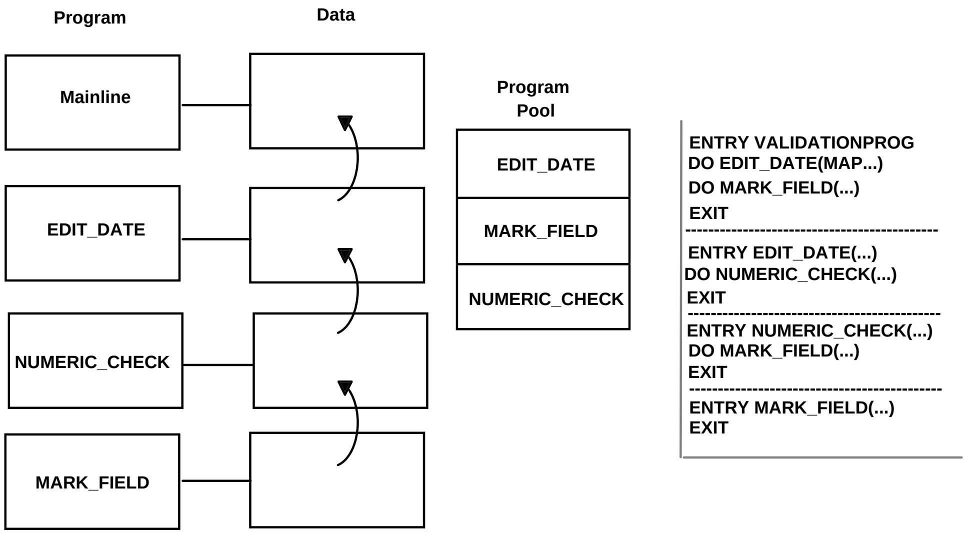


```
ENTRY VALIDATIONPROG
:
:
DO EDIT_DATE(MAP...)
:
:
EXIT
ENTRY EDIT_DATE(...)
:
:
DO NUMERIC_CHECK(...)
:
:
EXIT
ENTRY NUMERIC_CHECK(...)
:
:
DO MARK_FIELD(...)
:
:
EXIT
ENTRY MARK_FIELD(...)
:
:
EXIT
```

This is a mainline routine bound by ENTRY and EXIT statements. The DO statements indicate subroutines that reside physically with the calling program.

Execution involves one program work area, one data area, and one user word table. A maximum of 255 parameters is passed, but all parameters may be global. MANTIS stores the program as one entity.

The following figure presents an example of an external DO:



```
ENTRY VALIDATIONPROG
DO EDIT_DATE(MAP...)
DO MARK_FIELD(...)
EXIT
--------------------------------------------
ENTRY EDIT_DATE(...)
DO NUMERIC_CHECK(...)
EXIT
--------------------------------------------
ENTRY NUMERIC_CHECK(...)
DO MARK_FIELD(...)
EXIT
--------------------------------------------
ENTRY MARK_FIELD(...)
EXIT
```

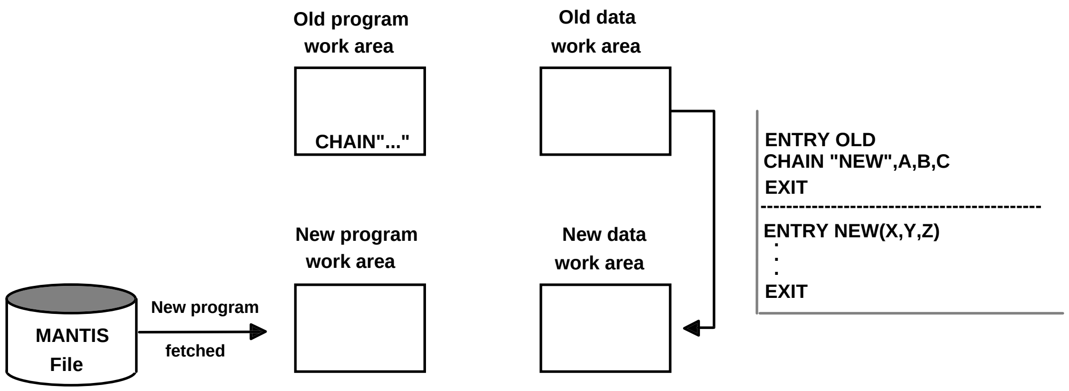
You must use ENTRY and EXIT statements around each physical program.  (Dashed lines indicate physical program boundaries.)

All user words are local to their own program.  Therefore, everything the subroutine uses from its caller must be defined on the ENTRY statement.  The parameter list of user words is by reference only.  No data is copied to an external data area.  A maximum of 255 parameters may be passed; variables not passed are local to each routine.  Each program is stored as a separate entity.

Each time a DO is executed, a new data area for the subroutine is allocated and rebuilt.  A program area is acquired and refreshed from the copy in memory (program pool).

External program code is optionally retained in the program pool when you EXIT.  External program data area is released at EXIT.

The following figure illustrates a program using the CHAIN statement:



ENTRY and EXIT statements are required around each physical program. (Dashed lines indicate physical program boundaries.)

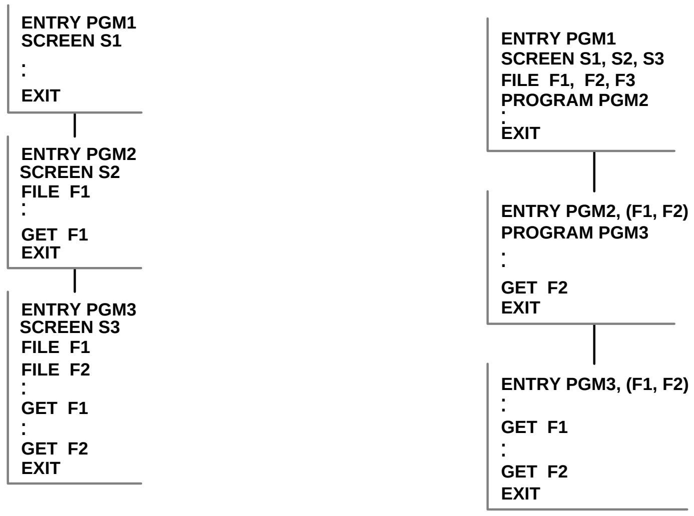
When MANTIS encounters a CHAIN statement, it loads the program from the cluster, and allocates and builds a new data work area. MANTIS then copies arguments from the old data work area into a new data work area. When it establishes new areas, it releases the old program and data work areas.

Referring to the application in the figure below (on the left side) as an example, examine how using various statements affects the number of disk accesses to the MANTIS cluster. CHAIN links PGM1, PGM2, and PGM3, and at each CHAIN, MANTIS fetches the new program from the cluster. Complex variables (e.g., SCREEN and FILE) must be defined in each new data work area.

If you use internal DO, you decrease the number of disk accesses because everything you execute is loaded once, which saves fetching programs and screens.

External DO (on the right in the figure below) enables you to define in the main program variables that are used in all routines. When defined in the main program, they can be passed to subsequent routines. For example, files F1, F2, and F3 are defined in PGM1, and can be passed to PGM2 and PGM3 with no disk access required. The only disk accesses required are those to fetch PGM2 and PGM3 the first time they are invoked.

**Disk Accesses using . . .**



**CHAIN or Internal DO**

```
ENTRY PGM1
SCREEN S1
:
EXIT

ENTRY PGM2
SCREEN S2
FILE  F1
:
GET  F1
EXIT

ENTRY PGM3
SCREEN S3
FILE  F1
FILE  F2
:
GET  F1
:
GET  F2
EXIT
```

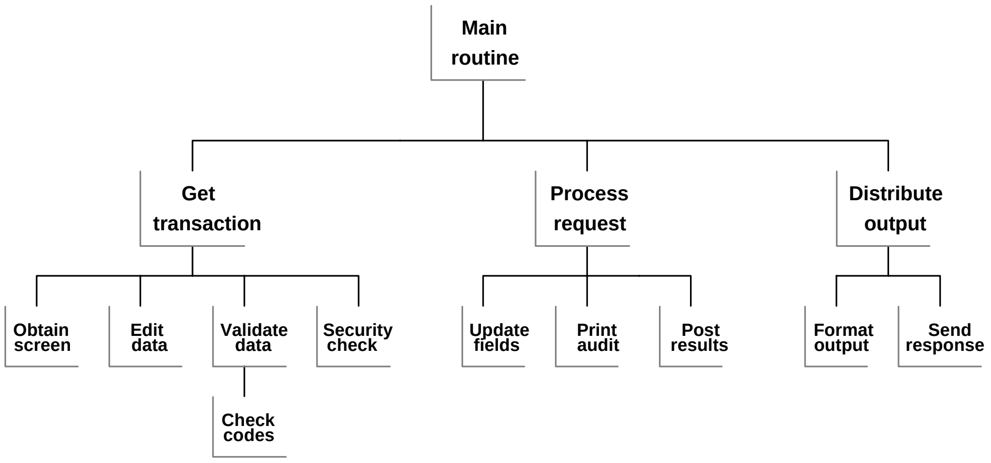
**External DO**

```
ENTRY PGM1
SCREEN S1, S2, S3
FILE  F1,  F2, F3
PROGRAM PGM2
:
EXIT

ENTRY PGM2, (F1, F2)
PROGRAM PGM3
:
GET  F2
EXIT

ENTRY PGM3, (F1, F2)
:
GET  F1
:
GET  F2
EXIT
```

## Modularizing

Many small externally done programs produce overhead. Each small program requires at least one disk access to the MANTIS cluster, plus extra processing and storage for each DO statement. To execute many small routines, group related routines into one program, then pass a parameter to the program indicating which routine should be executed. The main routine in the program will internally DO the subroutines.

In a system designed by way of a structured methodology, you can combine multiple related modules into one program. For example, the following figure shows how you can combine 14 separate modules into three programs, without making every module a separate program but rather mixing internally and externally done routines:



## Automatic mapping

To achieve automatic mapping of variables, define complex statements (SCREEN, FILE, etc.) at the same level, or pass them as arguments. Each subvariable is local to the routine where it is defined. This also applies to keys on INSERT and GET statements. The following examples show automatic mapping:

- When defining complex variables at one level, MANTIS automatically maps the fields in the record to be inserted from the screen. The CONVERSE can be in either SUB1 or the calling routine with the same results.

```
      10 ENTRY MAINLINE
      20 PROGRAM SUB1("SUB1", password)
      30 SCREEN MAP ("X"):  ¦ CONTAINS F1, F2, F3
      40 FILE (REC("Y","P"):  ¦ CONTAINS F1, F2, F3
      50 DO SUB1(MAP,REC)
automatically maps F1, F2, F3 between MAP and REC

      10 ENTRY SUB1(SC,FI)
      20 .CONVERSE SC
      30 .INSERT FI
40 EXIT
```

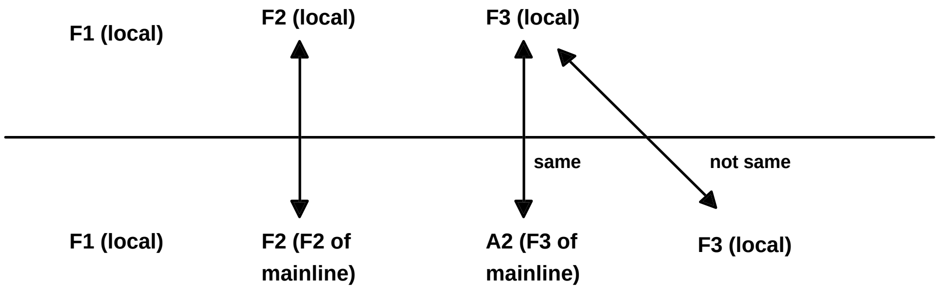
- In another example, F1, F2, and F3 are defined in the main routine. On entry to the subroutine, SC, F2, and A2 are defined as parameters in the subroutine. When the FILE statement executes, MANTIS looks for F1 and does not find it, so it defines one. When it encounters F2, it finds an F2 (because it appeared on the ENTRY statement) and connects to it. When F3 is encountered, MANTIS does not find it, so it defines another F3 (not the one in the main routine). A2 in the subroutine is the local name for the calling routine's F3 and shares its data area.

  The CONVERSE statement fills in all the variables in SC. The only variable automatically mapped in the CONVERSE and INSERT is F2.

Because F2 is defined as a parameter on the ENTRY statement, it will automatically be mapped between the file and the screen. Even though F3 is passed as an argument, the name “F3” is not known to the subroutine SUB1. Therefore, the FILE statement will establish a local variable F3 (as well as F1) in SUB1.

```
10 ENTRY MAINLINE
20 PROGRAM SUB1("SUB1", password)
30 SCREEN MAP ("X"):  ¦ CONTAINS F1, F2, F3
50 DO SUB1(MAP,REC)

10 ENTRY SUB1(SC,F2, A2)
20 FILE (REC("Y","P"):  ¦ CONTAINS F1, F2, F3
30 .CONVERSE SC
40 .INSERT FI
```

The following diagram shows what happens to the passed variables:



## Entity definition

Avoid defining entities within external programs, which causes recurring disk access to the MANTIS cluster and rebuilds the user word table. Declare all complex variables (SCREEN, FILE, ACCESS, etc.) as high up as possible in the hierarchy of external DO statements.  However, keep in mind the constraints for automatic mapping previously mentioned.

The data work area is released on EXIT and reacquired on each DO.  If you have complex statements in subroutines, they must be refetched from the cluster (causing disk accesses) and rebuilt (causing processor overhead) each time the program is done.  Even if you cannot define these entities at the highest level (DOLEVEL=0), the higher you can place these statements, the better.

## Frequency of DOs

The more often your application uses a subroutine, the more advantage you have using DO over a CHAIN statement (with a CHAIN to return) because the calling routine context is maintained, and the called routine does not have to be fetched from the cluster each time.  However, designing an external routine to do as much work as possible per call will provide better performance (e.g., if a routine can be called once to validate all elements of an array, it is better than calling repetitively for each element to be validated). Heavily used routines are best performed as internal routines.

## Debugging

Consider the following when debugging externally done programs:

♦ Use the UP and DOWN commands to move between the active DOLEVELs.  Use the “PROGRAM== >” heading to determine which program you are in.

♦ Use SHOW DOLEVEL to see which level is executing.  You can modify and replace programs at any level.  The revised version executes in the next run.

♦ Remember that arithmetic and text variables, lists or arrays should correspond in number between the ENTRY-EXIT and the DO or CHAIN statements.

## Program size

Moderate the size of your programs. Try not to go to either extreme (neither a few very large programs nor many small programs). If you have many large programs occupying memory as part of the hierarchy of external DOs, the amount of storage required to run MANTIS applications may increase.

## Programming considerations

With external DO, you can use the same FILE statement to reference many file descriptions because a new FILE statement is created for each call. The following two sample programs provide examples:

Program 1

```
          .
          PROGRAM FILENEW("PGM2","PASSPGM2")
          .
          .
          .
          .
          FILE_ID=LIBNAMEX
          INSERT_LEVEL=INSERT_PASSWORD
          DO FILENEW(FILE_ID,INSERT_LEVEL)
          .
          .
     .
```

Program 2

```
          ENTRY PGM2(FILE_NAME, PSSWD)
          .
          .
          .
          FILE F(FILE_NAME,PSSWD)
          .
          .
     .
```

You can change FILE_ID and INSERT_LEVEL in program 1 before any DO statement. MANTIS reexecutes the FILE statement for each call. This also works for other complex variable types (SCREEN, ACCESS, etc.).

## Performance

In externally done routines that will not be called again soon, add a RELEASE *program-name* to open more space in the program pool.  This reduces the amount of storage required by the application.  Note that the program must contain both PROGRAM and RELEASE statements.  If two or more programs have a PROGRAM statement for one program, any of the programs that have the PROGRAM statement can do the RELEASE, although the last program is preferable.  Good candidates for this are one-time routines (e.g., initialization, infrequent help) or routines that are infrequently called (e.g., setting profile information; routines called only at certain times, such as at the end of the month or the beginning of the day).

It is most appropriate to include RELEASE statements in calling programs that will run for long periods of time and not for programs that perform a quick task and CHAIN out.  Remember that all programs in the program pool are released automatically on a CHAIN or STOP. Therefore, doing a RELEASE just prior to a CHAIN or STOP is unnecessary.

The next external DO call to a program that has already been released will cause the program to be reloaded into the program pool where it will reside again until it is explicitly or implicitly released.  Appropriate logic for an infrequently used program can be similar to the following code sample:

```
IF choice
.PROGRAM INFR(pgmname,pgmpswd)
.DO INFR(p1,p2,p3)
.RELEASE INFR
END
```

## Miscellaneous programming hints

This section contains programming hints.

- When you edit or validate many fields on a screen, you normally want to highlight all the fields in error, issue a message, and position the cursor on the first error. The usual way to do this is to define a field (e.g., MESSAGE) and with each new error check, see if a previous error has occurred. For example:

```
         IF FIELD1_IN_ERROR
         .IF MESSAGE=""
         ..MESSAGE="FIELD1 IN ERROR"
         ..ATTRIBUTE(MAP,FIELD1)="BRI,CUR,RED"
         .ELSE
         ..ATTRIBUTE(MAP,FIELD1)="BRI,RED"
         .END
   END
```

  The processing logic can be simplified if you perform your edits from the bottom of the screen to the top. For example:

```
         WHEN FIELD10_IN_ERROR
         .MESSAGE="FIELD10 IN ERROR"
         .ATTRIBUTE(MAP,FIELD10)="BRI,CUR,RED"
         .
         .
         .
         WHEN FIELD9_IN_ERROR
         .MESSAGE="FIELD9 IN ERROR"
         .ATTRIBUTE(MAP,FIELD9)="BRI,CUR,RED"
         END
         .
         .
   .
```

  Using the previous method, the cursor is automatically positioned at the last named field (the top one in error) and the MESSAGE reflects the error for the last flagged field (the top one in error). This is even easier if there are vertical repeats. Then, only the subscript is decremented over the range and separate IF-END constructs are not required.

  This technique also can be used to check multiple, nonmutually-exclusive edits for one field.

- System testing of applications with date-dependent processing is easier if you determine the current date only once in the system (e.g., in the main menu program) and pass the date to application programs as a parameter.

- You can avoid hard coding your user name and password in each statement that calls for them by specifying them once at the beginning of the program.  For example:

  ```
  LIB="user-name:":LIBPASS="password"
  ```

  Later in your program, in a FILE statement, for instance, simply enter the following:

  ```
  FILE F(LIB+"filename",LIBPASS)
  ```

# 5

# Compatibility considerations (personal computer and IBM mainframe)

This manual describes the version of MANTIS that runs on Windows. You can design and execute your MANTIS applications on your personal computer. You can also transport your MANTIS applications from a personal computer to an IBM mainframe, using the Universal Export Facility (refer to the *MANTIS for Windows Facilities Reference Manual*, P19-2301).

Some differences exist between MANTIS for Windows and MANTIS for the IBM mainframe, resulting from the different internal architecture of the two products and the different hardware on which they run. For example, MANTIS on Windows allows you to have bigger programs with more variables than MANTIS for the IBM mainframe and is free of some of the constraints on screen layout imposed by the IBM 3270 display devices.

Compatibility mode enables you to write MANTIS applications on Windows that can run on the IBM mainframe. You can set compatibility mode in the Update Configuration File option (refer to the *MANTIS for Windows Administration Guide*, P19-2304). When testing your applications, MANTIS displays an error message whenever it detects a feature that is not supported by MANTIS for the IBM mainframe.

NOTE

All statements of a program must be executed to ensure that it is supported by MANTIS for the IBM mainframe.

This chapter describes the features that are affected by compatibility mode or need special consideration in applications that are to be migrated to the IBM mainframe.

# General considerations

This section describes compatibility considerations you should consider when writing your applications.

## Size limits

In MANTIS for Windows, various size limits can be specified by the Master User in the Update Configuration File option, but in compatibility mode the limits for MANTIS for the IBM mainframe are enforced.

| Size limit | Personal computer* | IBM mainframe |
|---|---|---|
| Maximum string length (bytes) | 254–32750 | 254 |
| Maximum dimension size (bytes) | 255–16000 | 255 |
| Maximum number of dimensions* | 2–255 | 2 (1 for TEXT) |
| Maximum program line number | 9999–64000 | 30000 |
| Maximum number of user words (variables) | 2048–65535 | 2048 |
| Maximum number of external DOLEVELS | 5–255 | 5 |
| Maximum number of CHAIN parameters | 40–255 | 40 |
| Maximum program size (bytes) | 32768–65248 | 32768 |

* Maximum size limits on the personal computer must be in the ranges listed here. The string maximum length subscript is counted in an array dimension count. Therefore, if the maximum number of dimensions option is 2, TEXT(15,4,16) is not allowed.

## Key assignments

MANTIS for Windows uses logical keys that correspond to 3270 keys. Logical keys are special key assignments used to enter data and perform special functions.  “Keyboard operation” on page 30 describes logical keys in more detail.  Personal computer keys are mapped to logical keys and this mapping can be customized (refer to the *MANTIS for Windows Administration Guide*, P19-2304).

## Indirect reference

Using the ampersand symbol (&) to make an indirect reference to a MANTIS symbolic name is not supported by MANTIS for the IBM mainframe.  MANTIS for Windows will return an error message if you try to use the ampersand in compatibility mode.

## Comments

MANTIS for Windows uses a broken vertical bar ( ¦ ) as the character to designate comments in a program, whereas MANTIS for the IBM mainframe uses a vertical bar (|) as the comment character.  When you migrate applications between these two environments, MANTIS automatically translates these characters for you.

## Kanji

MANTIS for Windows does not support Kanji (for Asian languages).  For compatibility with MANTIS for the IBM mainframe, you can define the data type of a field as K (for Kanji) in Screen, File, or External File Design.  However, you will receive an error if you try to use a Kanji field on MANTIS for Windows.

## Screen design

MANTIS for Windows supports a logical display area that is 32767 rows by 32767 columns. MANTIS for the IBM mainframe supports a logical display area that is 255 rows by 255 columns. In MANTIS for the IBM mainframe, you can design a screen as large as 255 rows by 255 columns. In MANTIS for Windows, you cannot design a screen as large as 255 rows by 255 columns because of the 64K limitation on the overall screen size.

Windowing in MANTIS for Windows is restricted to the size of the current map set (you can scroll only to the edge of the map set). Windowing in MANTIS for the IBM mainframe is restricted to the size of the logical display (you can scroll anywhere in the logical display area, regardless of the map set size).

MANTIS for the IBM mainframe does not support AUTOMATIC WINDOWING, which is the default in MANTIS for Windows. Specify Y (yes) for this attribute in the Library Functions option of Screen Design if you want MANTIS for Windows to treat partially displayed fields in the same way as MANTIS for the IBM mainframe. However, when automatic windowing is disabled (by specifying N), you will be prevented from updating a window-clipped (partially displayed) field on screen, even when the map is conversed with the UPDATE option. This is not compatible with MANTIS for the IBM mainframe.

MANTIS for Windows uses a broken vertical bar (¦) as the default blank-fill character, whereas MANTIS for the IBM mainframe uses a solid vertical bar (|).

The Valid Name, Validation List, and Variable Extended Edit attributes in Screen Design are not IBM compatible.

## ASCII and EBCDIC collating sequences

In MANTIS for Windows, text values are stored using the ASCII code for each character, whereas MANTIS for the IBM mainframe uses the EBCDIC code for each character. The different numerical values of ASCII and EBCDIC codes cause different collating sequences. Comparison of text values and order of file keys can produce different results in MANTIS for Windows and MANTIS for the IBM mainframe, except when testing for equality.

## Field input

In MANTIS for Windows, any changes made to screen fields are automatically reflected in the corresponding program variables. In MANTIS for the IBM mainframe, whether fields are updated depends on the key pressed. Fields are not updated on the mainframe unless ENTER or a PF key is pressed.

## Noninteractive input

Using /STDIN: to direct MANTIS to obtain keyboard input from a file is not supported by MANTIS for the IBM mainframe. The format of noninteractive input is different between MANTIS for Windows and Batch MANTIS on the mainframe, and cannot be transported between the two environments.

## Mainframe to Windows

The following list provides the statements and functions that are available in MANTIS for the IBM mainframe, but are not supported by MANTIS for Windows:

| | |
|---|---|
| ASI* | KANJI* |
| CALL* | MARK* |
| DEQUEUE** | RESET** |
| ENQUEUE** | TOTAL* |
| FSI* | VIEW* |
| INTERFACE* | VSI* |

* MANTIS will display an error message if you try to execute this statement or function on Windows.

** This statement is simply ignored on MANTIS for Windows.

## Programming statements and functions

The following statements and functions are affected by compatibility mode or need special consideration in applications that are to be migrated to the IBM mainframe.

### ACCESS statement

The NEW, REPLACE, and FILE parameters of the ACCESS statement are not supported by MANTIS for the IBM mainframe.  MANTIS for Windows will return an error message if you try to execute an ACCESS statement with a NEW, REPLACE, or FILE parameter in compatibility mode.

### ATTRIBUTE statement

MANTIS for the IBM mainframe, the PRINTER parameter of the ATTRIBUTE statement is restricted to the Master User library.

The following attributes are ignored in MANTIS for Windows, but take effect in programs that are executed in MANTIS for the IBM mainframe:

♦ DETECTABLE/NON DETECTABLE

♦ OVERLINE/NO OVERLINE

♦ LEFT BAR/NO LEFT BAR

♦ RIGHT BAR/NO RIGHT BAR

♦ MODIFIED/UNMODIFIED

The UPPERCASE and LOWERCASE attributes have an effect in MANTIS for Windows, but are not supported in programs that are executed in MANTIS for the IBM mainframe.

The CLASS attribute has a different interpretation in MANTIS for Windows and MANTIS for the IBM mainframe.  See “ATTRIBUTE” on page 81 for more information.

The SPOOL ON CLOSE attribute is not supported by MANTIS for the IBM mainframe.

## BREAK statement

The BREAK statement is supported in Releases 5.3 and above of MANTIS for the IBM mainframe.  MANTIS for Windows allows the BREAK statement in compatibility mode.

## CHAIN statement

Arguments that are expressions are not supported by of MANTIS for the IBM mainframe.  MANTIS for Windows will return an error message if you try to execute a CHAIN statement with arguments that are expressions in compatibility mode.

The LEVEL parameter is supported by Releases 5.2 and above of MANTIS for the IBM mainframe.  The number of arguments passed on the CHAIN statement must be equal to the number of formal arguments on the ENTRY statement of the chained-to program.  Note that the Master User does not have this restriction.

## CHR function

The CHR function is supported by Releases 5.4 and above of MANTIS for the IBM mainframe.  MANTIS for Windows will allow a CHR function in compatibility mode.

## CLEAR statement

MANTIS for Windows will return an error message if you try to execute a CLEAR statement with any parameter other than a screen name in compatibility mode.

## COMMIT statement

The ON and OFF parameters are supported by Releases 5.3 and above of MANTIS for the IBM mainframe.  MANTIS for Windows allows COMMIT ON and COMMIT OFF in compatibility mode.

## CONVERSE statement

In MANTIS for the IBM mainframe, the WINDOW parameter invokes window mode in which the program function keys are used for windowing functions instead of their normal purpose.  Window mode is terminated by the completion of CONVERSE processing or by pressing PF9/21.

In MANTIS for Windows, window mode is always in effect.  Windowing functions can always be performed by using the logical windowing keys.

In MANTIS for Windows, the only difference between the WINDOW and DISPLAY parameters on the CONVERSE statement is that the row and column coordinates of the window are displayed if WINDOW is specified.

The RELEASE parameter is not supported by Releases 5.2 and above of MANTIS for the IBM mainframe.

The template parameter is not supported by MANTIS for the IBM mainframe.

## DO statement

Arguments that are expressions are not supported by MANTIS for the IBM mainframe.  For external subprograms, arguments must be defined in IBM mode.  In an IBM environment, you may have an undefined parameter on a DO statement.  It defaults to BIG.  If you have an undefined parameter on a DO statement for an external subroutine, an error message is issued.  MANTIS for Windows will return an error message if you try to execute a DO statement with arguments that are expressions in compatibility mode.

## EDIT statement

The EDIT statement is not supported by MANTIS for the IBM mainframe. MANTIS for Windows will return an error message if you try to execute an EDIT statement in compatibility mode.

## FOR statement

The FOR statement is supported by Releases 5.2 and above of MANTIS for the IBM mainframe.  MANTIS for Windows allows a FOR statement in compatibility mode.

## FSI function

In MANTIS for Windows, the additional status values of ERROR and TIMEOUT can be returned by the *interface-name* function. See “CALL” on page 90 for more information.

## GET statement

The ENQUEUE parameter of the GET statement is allowed for MANTIS files or external files in Releases 5.2 and above of MANTIS for the IBM mainframe. MANTIS for Windows allows a GET statement with an ENQUEUE parameter for a MANTIS or external file in compatibility mode.

In MANTIS for the IBM mainframe, the NEXT parameter of the GET statement is not allowed when a key is also specified for MANTIS files or external files. MANTIS for Windows will return an error message if you try to execute a GET statement with a NEXT parameter when a key is also specified for a MANTIS or external file in compatibility mode.

## KEY function

In MANTIS for Windows, the KEY function returns a text value identifying your response to the most recent CONVERSE, OBTAIN, PROMPT, or WAIT statement. For CONVERSE, the value is the name of the action or macro key you entered (see the tables under “MANTIS action keys” on page 35 and “MANTIS macro keys” on page 36). For OBTAIN, PROMPT, and WAIT, this value is replaced by the contents of the Key Simulation field. In MANTIS for the IBM mainframe, the KEY function always returns the actual key pressed.

## LANGUAGE function

The LANGUAGE function is not supported by MANTIS for the IBM mainframe. MANTIS for Windows will return an error message if you try to execute a LANGUAGE function in compatibility mode.

## NEXT statement

The NEXT statement is supported by Releases 5.3 and above of MANTIS for the IBM mainframe. MANTIS for Windows allows a NEXT statement in compatibility mode.

## NULL function

The NULL function is supported by Releases 5.2 and above of MANTIS for the IBM mainframe. MANTIS for Windows allows a NULL function in compatibility mode.

## NUMERIC function

The NUMERIC function is supported by Releases 5.3 and above of MANTIS for the IBM mainframe. MANTIS for Windows allows the NUMERIC function to be used in compatibility mode.

## OUTPUT statement

MANTIS for the IBM mainframe gives control to a printer exit if the VIA exit parameter of the OUTPUT statement is used. In MANTIS for Windows, the VIA exit parameter is ignored.

## PERFORM statement

In MANTIS for Windows, the parameter of the PERFORM statement is a DOS command that can perform any DOS function, including running a program. In MANTIS for the IBM mainframe, the parameter of the PERFORM statement is the name of the program or transaction to be performed (with other optional information).

In MANTIS for Windows, you can use a dollar sign ($) instead of a PERFORM command. This abbreviation is not supported by MANTIS for the IBM mainframe.

## PRINTER= statement

In MANTIS for Windows, the parameter of the PRINTER= statement is either the device name of the printer or a file name to which printed output will be written. In MANTIS for the IBM mainframe, the parameter of the PRINTER= statement is a CICS or VSAM identifier. These parameters are not meaningful in programs migrated from Windows to the IBM mainframe. It is recommended that applications to be migrated from Windows to the IBM mainframe do not use the PRINTER= statement and send their printed output to the default printer defined in the user profile.

Unlike MANTIS for the IBM mainframe, in MANTIS for Windows PRINTER= closes the current printer output and does an implicit OUTPUT SCREEN statement.

## RELEASE statement

The *access-name* or *interface-name* parameter is not supported by MANTIS for the IBM mainframe. MANTIS for Windows will return an error message if you try to execute a RELEASE statement with an *access-name* or *interface-name* parameter in compatibility mode.

## RETURN statement

The RETURN statement is supported by Releases 5.3 and above of MANTIS for the IBM mainframe. MANTIS for Windows allows a RETURN statement in compatibility mode.

## SCROLL statement

In MANTIS for the IBM mainframe, the ON and OFF parameters of the SCROLL statement select the scroll mode of the terminal, but MANTIS for Windows ignores these parameters.

## SET statement

The SET statement is not supported by MANTIS for the IBM mainframe. MANTIS for Windows will return an error message if you try to execute a SET statement in compatibility mode.

## SHOW statement

In compatibility mode, data items must be separated by a comma (,) or semicolon (;). MANTIS for Windows will return an error message if you try to execute a SHOW statement with no separators between data items in compatibility mode. When not in compatibility mode, the separator can be omitted to display the data items without intervening spaces.

In compatibility mode, a comma or semicolon must be used as a separator after AT column, but does not affect the column position. When not in compatibility mode, a comma or semicolon after AT column advances to the next screen zone or column position.

## SLICE and SLOT statements

In MANTIS for the IBM mainframe, the parameters of these statements affect system performance, but in MANTIS for Windows they do not.

In MANTIS for Windows, do not use SLICE and SLOT as debugging devices for single stepping as you can in MANTIS for the IBM mainframe.

## $SYMBOL function

The $SYMBOL function is not supported by MANTIS for the IBM mainframe. MANTIS for Windows will return an error message if you try to execute a $SYMBOL function in compatibility mode.

## UPPERCASE function

The UPPERCASE function is supported by Releases 5.2 and above or MANTIS for the IBM mainframe. MANTIS for Windows allows an UPPERCASE function in compatibility mode.

# A

# Dissimilarity debugging

Dissimilarity errors occur in MANTIS when processing a complex statement (SCREEN, FILE, or ACCESS). This appendix defines the types of dissimilarity and suggests how you may locate and correct these errors.

Errors can occur because of dissimilarity in the following:

- **Type**. Occurs when a field defined in the specified design or view is already defined in the program, but with an incompatible type (TEXT, BIG, or SMALL).
- **Dimension**. Occurs when a field defined in the specified design or view is already defined in the program, but with different dimensions (or no dimensions), or implied by the number of *levels* specified on the complex statement (in the case of FILE and ACCESS).

MANTIS displays the name of the field in question in the error message. Use the DISPLAY command (refer to the *MANTIS for Windows Facilities Reference Manual*, P19-2301, to determine its type and dimensions, and the USAGE command to find the previous definition). Note that the previous definition may be by another complex statement, in which case the USAGE command will not display it. Also, if the variable was passed to the routine through an ENTRY statement, you may have to work backwards into the calling routine to resolve the problem. Finally, verify that the field definition in the design or view and, if applicable, the number of *levels* specified, are correct.

# B

# Messages

This appendix contains the messages you may receive while using the MANTIS programming language or running a MANTIS program. Messages are listed in numerical order by error code.

**075** `The line number is not in the allowed range.`

**Explanation** The allowed range for program line numbers is from 1 to the maximum line number set through the configuration file.

**Action** Use a line number in the allowed range, or use the Update Configuration File option in the Master User Facility to increase the maximum line number.

**076** `The line is too long or too complex.`

**Explanation** MANTIS could not encode the line without exceeding the maximum allowable encoded line length.

**Action** Break the line down into separate statements on separate lines, or use continuation lines to break up a long statement.

**077** `The magnitude of the number is not in the allowed range.`

**Explanation** MANTIS could not convert the number to floating-point format, because the exponent would exceed the allowable range. MANTIS only accepts numbers whose magnitude is between 1e-308 and 1e+308.

**Action** Rework your program to avoid using out-of-range values.

**078** `The closing quote is missing from a text literal.`

**Explanation** A text literal must be enclosed in quotation marks (e.g., “HELP”).

**Action** Insert the missing quotation mark.

**080** `The line contains a character which is not in the MANTIS character set.`

**Explanation** The MANTIS character set is discussed in “Character set” on page 40. Special characters are listed in the table in that section. Any other characters are allowed only in text literals or in comments.

**Action** Restrict your MANTIS statements and commands to the MANTIS character set.

**081** `The ROUNDED parameter is not in the correct format.`

**Explanation** ROUNDED can be followed only by an equal sign or by an integer from 0–6 in parentheses (e.g., X ROUNDED(3) = Y).

**Action** Correct the ROUNDED parameter to comply with the above format.

**083** `A logic keyword is not at the start of the line.`

**Explanation** A statement with a logic keyword must appear on a line by itself. MANTIS logic keywords are as follows:

| | | |
|---|---|---|
| BREAK | CHAIN | DO |
| ELSE | END | ENTRY |
| EXEC SQL | EXIT | FOR |
| IF | NEXT | RETURN |
| UNTIL | WHEN | WHILE |

**Action** Put the statement with a logic keyword on a separate line.

**084** `The line contains a keyword which can be used only as a command.`

**Explanation** The following keywords can only be used as commands (not as statements in your program):

| | | |
|---|---|---|
| ALTER | CHANGE | COPY |
| DISPLAY | DOWN | ERASE |
| GO | HELP | LIST |
| LOAD | NEW | PURGE |
| QUIT | REPLACE | RUN |
| SAVE | SEQUENCE | UP |
| USAGE | | |

**Action** Remove the command keyword from your program line.

**085** `A symbolic name is too long.`

**Explanation** The symbolic name exceeds the maximum allowable size of 80 characters.

**Action** Reduce the length of the symbolic name.

**086** `The line contains a keyword which can be used only as a statement.`

**Explanation** The following keywords can be used only as program statements (not as commands):

| | | |
|---|---|---|
| BREAK | CHAIN | DO (internal subroutine) |
| ELSE | END | ENTRY |
| EXEC SQL | EXIT | FOR |
| IF | NEXT | RETURN |
| UNTIL | WHEN | WHILE |

**Action** Remove the statement keyword from your command line.

**088** `MIXED character strings are not valid in compatibility mode.`

**Explanation** Literals containing both text and Kanji characters are not supported in compatibility mode.

**Action** Change the program, or use the Update Configuration File option in the Master User Facility to change the compatibility mode setting.

**089** `Screens without separators are not supported in compatibility mode.`

**Explanation** When running in compatibility mode, a SCREEN statement was used that references a screen design without field separators.

**Action** Change the screen design or use the Update Configuration File option in the Master User Facility to change the compatibility mode setting.

**090** `There is no such line for deletion.`

**Explanation** When you enter a line number by itself, MANTIS deletes that line.  In this case, there is no line having the number you entered.

**Action** Check the number of the line you meant to delete, or put something after the line number if you meant to insert or replace the line.

**091** `The program is too big.`

**Explanation** In MANTIS internal format, your program exceeds the maximum size in bytes set through the configuration file.

**Action** Restructure your program by use of external subroutines, or use the Update Configuration File option in the Master User Facility to increase the maximum program size.

**093** `The DO command cannot be used to execute an internal subroutine.`

**Explanation** Although the DO keyword can be used as a command to execute an external subroutine, it can be used only as a statement to execute an internal subroutine. An internal subroutine is part of the current program, whereas an external subroutine is in a separate program identified by a PROGRAM statement.

**Action** Execute the internal subroutine using a DO statement in your program.

**094** `A program line number is too big.`

**Explanation** The original default maximum line number is 64000, but you may have lowered this limit through the configuration file.

**Action** Use the SEQUENCE command to resequence your program with a smaller increment between lines, or use the Update Configuration File option in the Master User Facility to increase the maximum program line number.

**095** `The statement has too many parameters.`

**Explanation** A maximum of 255 parameters may be specified for the following MANTIS statements:

| | | |
|---|---|---|
| ATTRIBUTE | CHAIN | CLEAR |
| DO | GET | LET |
| OBTAIN | SHOW | |

**Action** Correct the number of parameters in your statement.

**100** `An expression contains an invalid operand.`

**Explanation** An operand in an arithmetic expression must be a numeric variable, numeric array element, numeric constant, built-in numeric function, built-in numeric constant, or subexpression in parentheses. An operand in a text expression must be a text variable, a text array element, a text substring, a text literal, a built-in text function, or a subexpression in parentheses.

**Action** Remove the invalid operand from the expression.

**101** `An expression contains incompatible operands.`

**Explanation** The operands on either side of an operator are incompatible. BIG and SMALL operands are compatible with each other, but TEXT and KANJI operands are not compatible with any other types, except for the NULL string, where data type is not considered.

**Action** Make the operands compatible (e.g., by using a built-in function):

- ♦ TXT(A) returns the text value of the arithmetic expression A.
- ♦ VALUE(T) returns the numeric value of the text expression T.

**102** `An expression contains an invalid operator.`

**Explanation** Arithmetic operators used with numeric operands are: **, *, /, +, -. Text operators used with text operands are: +, -. Relational operators, used with numeric or text operands, are: =, <, >, <=, >=, <>. Logical operators, used with logic values, are: AND, OR.

**Action** Use only the operators appropriate for the operands in the expression.

**103** `The string stack is too small for a text expression to be evaluated.`

**Explanation** MANTIS uses a string stack while evaluating text expressions. A text value in the current expression will not fit on the stack. The length of the string evaluation stack is set through the configuration file.

**Action** Use simpler text expressions, or use the Update Configuration File option in the Master User Facility to increase the string evaluation stack length.

**104** `An expression contains an operand of an invalid data type.`

**Explanation** One of the operands in an expression is not a valid operand data type.

**Action** This error indicates an internal MANTIS error. Please report the problem to your CINCOM representative.

**105** `The NULL text function is not supported in compatibility mode.`

**Explanation** Compatibility mode has been set to ensure that the programs you write can also run on an IBM mainframe. In compatibility mode, features that are not supported by MANTIS for the IBM mainframe cannot be used.

**Action** Avoid using the NULL text function (e.g., by using the empty string “”) or use the Update Configuration File option in the Master User Facility to turn off compatibility mode.

**106** `A left parenthesis is missing.`

**Explanation** Parentheses are required after an array reference within an expression. They are also required to enclose parameters on many MANTIS statements.

**Action** Insert the missing parenthesis.

**107** `A right parenthesis is missing.`

**Explanation** A right parenthesis is needed to match the preceding left parenthesis. You may have included too many parameters before the right parenthesis.

**Action** Insert the missing parenthesis, or remove superfluous parameters.

**108** `The argument of a built-in function is not a text expression.`

**Explanation** The built-in functions SIZE, VALUE, POINT, and NUMERIC require their argument to be a text expression.

**Action** If you want to use one of these functions, make the argument a text expression.

**109** `The argument of a VALUE function cannot be a KANJI string.`

**Explanation** MANTIS cannot translate KANJI strings to numeric values. Only a TEXT string can be converted into a number by the VALUE function.

**Action** Change the argument of VALUE to a TEXT string.

**110** `The argument of a built-in function is not an arithmetic expression.`

**Explanation** The arguments of the following built-in functions must be arithmetic expressions:

| | | | |
|---|---|---|---|
| ABS | ATN | COS | EXP |
| INT | LOG | NOT | RND |
| SGN | SIN | SQR | TAN |
| TXT | | | |

**Action** Make the argument an arithmetic expression.

**114** `The unary minus operator can be used only with a numeric operand.`

**Explanation** The unary minus operator cannot precede a text operand in an expression.

**Action** Do not use a unary minus with a text operand.

**115** `An array element has too many subscripts.`

**Explanation** The number of subscripts of an array element must not exceed the number of dimensions specified when the array was defined.

**Action** Change the array element or the array definition to reconcile the number of subscripts with the number of dimensions.

**116** `A subscript of an array element is not an arithmetic expression.`

**Explanation** Each subscript of an array element must be an arithmetic expression.

**Action** Make sure that each array element subscript is an arithmetic expression.

**117** `An array element subscript exceeds the corresponding dimension.`

**Explanation** Each subscript of an array element must have a value less than or equal to the corresponding dimension specified when the array was defined.

**Action** Reduce the size of the subscript or increase the dimension of the array.

**118** `The start or end position of a substring is not an arithmetic expression.`

**Explanation** A text substring is referenced by start and end character positions within a text variable or array element. Character positions are numbered 1,2,3,... from left to right in the text string, or -1,-2,-3,... from right to left in the text string. The start and end positions are each specified by an arithmetic expression.

**Action** Specify the start and end positions (using either numbering scheme) by means of arithmetic expressions.

**119** `Start or end position of a substring exceeds the maximum string length.`

**Explanation** A text substring is referenced by start and end character positions within a text variable or array element. The character positions are numbered 1,2,3,... from left to right in the text string, or -1,-2,-3,... from right to left in the text string. The absolute value of the start and end position cannot exceed the maximum string length set by your Master User.

**Action** Specify start and end positions that are within the maximum string length.

**120** `An array element has no subscripts.`

**Explanation** A reference to a symbolic name that has been defined as an array must be followed by at least one subscript in parentheses.

**Action** Insert subscript(s) if you meant the symbolic name to be an array, otherwise remove the dimension(s) from the definition of the symbolic name.

**121** `An expression is incomplete.`

**Explanation** The evaluation of an expression was incomplete when the end of the line was reached, or a colon (statement separator) was encountered.

**Action** Correct the incomplete expression.

**122** `Symbolic name exceeds the maximum number of dimensions or string length.`

**Explanation** The definition of the symbolic name exceeds the maximum number of dimensions or the maximum string length set through the configuration file.

**Action** Reduce the length or the number of dimensions, or use the Update Configuration File option in the Master User Facility to increase the limits.

**123** `A negative base of a Power (**) operator requires an integral exponent.`

**Explanation** A negative number cannot be raised to a fractional power.  For example: -2 ** 3.5 is an invalid operation because -2 is a negative base and 3.5 is not an integral exponent.

**Action** If you intended this sort of operation, first convert the base to its absolute value and handle the sign of the result according to your own rules.

**124** `Invalid statement in Run-Only system.`

**Explanation** The MANTIS you are using is a run-only (nondevelopment) version, and the statement you tried to execute is only valid in a full developer’s version of MANTIS.

**Action** Either remove the statement or use a full version of MANTIS.

**125** `The LANGUAGE function is not supported in compatibility mode.`

**Explanation** MANTIS for the IBM mainframe does not support the LANGUAGE function.

**Action** Do not use the LANGUAGE function, or use the Update Configuration File option in the Master User Facility to change the compatibility mode setting.

**129** `Subscripts can be used only with array, string, or complex variables.`

**Explanation** The subscripted variable reference is not valid. Subscripts can be specified only with numeric array variables, scalar or array string variables, or complex variables. Subscripting a scalar string variable or a complex variable is a substringing operation.

**Action** Verify that the subscripted variable has been correctly defined by an explicit (BIG, TEXT, etc.) or implicit (SCREEN, FILE, etc.) declaration.

**130** `The message parameter is not the symbolic name of a text array.`

**Explanation** This is an internal error.

**Action** Contact your Cincom representative.

**131** `The length parameter is not the symbolic name of a numeric variable.`

**Explanation** This is an internal error.

**Action** Contact your Cincom representative.

**133** `Continuation of a comment is not allowed.`

**Explanation** The previous line ended with a comment (|) followed by comment text, and the current line begins with a continuation character ('). Comments cannot be continued in this way.

**Action** Replace the continuation character with a comment character to make the current line into a comment.

**134** `The message parameter is incorrectly dimensioned.`

**Explanation** This in an internal error.

**Action** Contact your Cincom representative.

**150** `The allowed number of variables has been exceeded.`

**Explanation** You attempted to define a variable with an internal “user word number” greater than the maximum user word number set through the configuration file.

**Action** Reduce the number of variables in your program, or use the Update Configuration File option in the Master User Facility to increase the maximum user word number.

**151** `The ENTRY statement cannot be used inside a subroutine or logic block.`

**Explanation** A subroutine is defined by enclosing a block of statements within ENTRY-EXIT statements. Subroutine definitions cannot be nested; each subroutine must be terminated by an EXIT statement before another subroutine can be started with an ENTRY statement. A subroutine definition cannot be started inside a FOR-END, IF-ELSE-END, WHEN-END, WHILE-END, or UNTIL-END logic block.

**Action** Rework your subroutine definition without nesting.

**152** `The ELSE statement has no matching IF statement.`

**Explanation** The ELSE statement can be used only as part of an IF-ELSE-END sequence of statements.

**Action** Rework your program logic.

**153** `The END statement has no matching logic statement.`

**Explanation** The END statement is not matched by any of the following logic statements:

| | | |
|---|---|---|
| EXEC SQL | FOR | IF/ELSE |
| UNTIL | WHEN | WHILE |

**Action** Check for too many END statements prior to this one in the program.

| | | |
|---|---|---|
| **154** | `The EXIT statement has no matching ENTRY statement.` | |
| | **Explanation** | The EXIT statement can be used only as part of an ENTRY-EXIT sequence of statements to define a subroutine. |
| | **Action** | Rework your subroutine definitions. |
| **156** | `An END or EXIT statement is missing.` | |
| | **Explanation** | An END statement is required to terminate a FOR-END, IF-ELSE-END, WHEN-END, WHILE-END, or UNTIL-END sequence of statements, or an EXIT statement is required to terminate an ENTRY-EXIT subroutine definition. |
| | **Action** | Rework your program logic or subroutine definitions. |
| **157** | `The libname parameter is incorrectly specified.` | |
| | **Explanation** | The *libname* parameter specifies the name of an entity in a user's library in the format [*user name*:]*entity-name*. The user name cannot exceed 16 characters. The *entity-name* cannot exceed 30 characters. |
| | **Action** | Correct the *libname* parameter. |
| **158** | `There is no such user defined to MANTIS.` | |
| | **Explanation** | The user name in a *libname* parameter must be a user name as defined through the Master User facilities. The Master User facilities are used to create a profile for each MANTIS user. |
| | **Action** | Respecify the user name. |
| **159** | `The User Profile contains an invalid user code.` | |
| | **Explanation** | When a User Profile is created for a MANTIS user, MANTIS assigns an internal user code and stores it in the User Profile. The profile for the user name specified in the *libname* parameter contains a code that is not in the correct range. |
| | **Action** | Inform your Master User. |

**160** `No more storage space can be obtained for symbolic names.`

**Explanation** MANTIS stores symbolic names using a vocabulary area and an index area.  These areas are expanded incrementally as required by requesting storage space from Windows.  The increment size for the storage request is set through the configuration file.  Windows could not provide the increment requested.

**Action** Use your Master User facilities to reduce the vocabulary area increment size and/or the index area increment size.  If this doesn’t work, provide more storage to MANTIS by closing other applications or by installing more memory in your personal computer.

**161** `The profile is corrupt.`

**Explanation** When MANTIS read the profile for the specified file, screen, or interface, the actual length of the profile was inconsistent with its stated length.  It may have been corrupted as a result of a system failure.

**Action** Delete the profile, using the appropriate Library Facility, and recreate it.

**162** `The program contains unresolved BREAK/NEXT statement(s).`

**Explanation** The program contains BREAK and/or NEXT statements that are not within the scope of any FOR, WHEN, WHILE, or UNTIL statement block.

**Action** Check the program logic carefully.  List the program and note the indentation level of the BREAK/NEXT statement(s).  Check that there is at least one of the above loop statements with a lower indentation level.

**170** `The resource name is too long.`

**Explanation** The resource name parameter, which must be a text expression, cannot exceed 31 characters.

**Action** Change the name so it does not exceed 31 characters.

**183** `The File Access method is not supported.`

**Explanation** You have attempted to execute an ACCESS statement for a File View for PC Contact. This is not permitted.

**Action** Change the File View to one that is supported by MANTIS for Windows.

**184** `The TOTAL DBMS product is not installed on this system.`

**Explanation** MANTIS for Windows cannot access TOTAL files.

**Action** Remove or comment out the TOTAL statement.

**185** `The RDM product is not installed on this system.`

**Explanation** MANTIS for Windows cannot access RDM views.

**Action** Remove or comment out the VIEW statement.

**200** `Record retrieval failed.`

**Explanation** MANTIS received an error status when attempting to read a record from a MANTIS file or external file.

**Action** For a MANTIS or external file, obtain diagnostic information by using the HELP LAST command to display the last message. Additional information may also be obtained from the MANTIS error log file.

**203** `The libname parameter is not a valid text expression.`

**Explanation** The *libname* parameter must be a text expression that evaluates to a text value of not more than 47 characters. This convention allows for a 16-character user name, a colon, and a 30-character entity name.

**Action** Correct the *libname* parameter.

**204** `A comma is missing.`

**Explanation** A comma is used to separate parameters in a statement or arguments in a function. In this case, a parameter or argument is not followed by a comma.

**Action** Insert the missing comma.

**205** `The password parameter is not a valid text expression.`

**Explanation** The password parameter must be a text expression that evaluates to a text value of no more than 16 characters.

**Action** Correct the password parameter.

**206** `The levels parameter does not have a value in the range 0–255.`

**Explanation** The *levels* parameter must be an arithmetic expression that evaluates to a value in the range of 0–255. Specifying a value of zero is equivalent to omitting the *levels* parameter.

**Action** Correct the *levels* parameter.

**208** `There is no profile with the name specified in the libname parameter.`

**Explanation** The *libname* parameter specifies an entity name preceded by a colon and a user name if another user's library is to be searched for the entity (your own library is the default). The entity with the name you specified is not in the library specified. You can use the Directory Facility to show the entities in your own library.

**Action** Correct the *libname* parameter.

**209** `Too many Record Layout profiles are chained together.`

**Explanation** In File Design, you can specify an Associated Record Layout if the file profile has the same record layout as another profile. The specified profile can in turn specify an Associated Record Layout, and so on. After 16 successive Associated Record Layout profiles, however, MANTIS assumes that something is wrong.

**Action** Check that the chain of profiles does not loop back on itself.

**210** `The Associated Record Layout profile does not exist.`

**Explanation** In File Design, you can specify an Associated Record Layout if the file profile has the same record layout as another profile. When a profile is loaded to process a FILE statement, the specified Associated Record Layout profile must exist in the library.

**Action** Use File Design to correct the profile specification.

**211** `The profile status is not ‘ACTIVE’.`

**Explanation** File Design and External File View Design allow you to specify the status of each profile. If anything other than ACTIVE is entered in the status field, your program is prevented from accessing the profile.

**Action** Change the status to ACTIVE if you want your program to access the profile.

**212** `Type or dimensions in profile are inconsistent with an existing variable.`

**Explanation** During processing of an ACCESS, FILE, or SCREEN statement, variables and arrays are defined in your work area according to the profile specified in the corresponding Design Facility. When variables or arrays from different profiles have the same name, MANTIS assumes that this is intended to affect the automatic transfer of data between entities (e.g., between a screen and a file record). MANTIS accordingly checks the type and dimensions for consistency.

**Action** Use the DISPLAY ALL command to display the attributes of all the variables and arrays in your work area. This will help you to identify the conflicting names or attributes.

**213** `A MANTIS file or external file could not be opened.`

**Explanation** MANTIS received an error status when attempting to open the file.

**Action** Obtain diagnostic information by using the HELP LAST command to display the last error message. Additional information may also be obtained from the MANTIS error log file.

**214** `The symbolic name was not defined by FILE or ACCESS.`

**Explanation** A GET, UPDATE, INSERT, or DELETE keyword must be followed by the symbolic name of a MANTIS file or external file, as defined in a FILE or ACCESS statement.

**Action** Supply the correct symbolic name.

**215** `You are not authorized to perform this operation.`

**Explanation** For a MANTIS file or an external file, the access authority is determined by the password specified in the corresponding FILE or ACCESS statement. MANTIS compares this password with the passwords specified in the profile by the corresponding design facility, and sets the access authority accordingly.

**Action** Use the correct password.

**216** `A record in the MANTIS file is corrupt.`

**Explanation** When MANTIS read the record from the MANTIS file, the actual length of the record was inconsistent with its stated length. The record may have been corrupted as a result of a system failure.

**Action** Delete the record from the MANTIS file and recreate it.

**217** `Record insertion failed.`

**Explanation** MANTIS received an error status when attempting to insert a record into a MANTIS file or external file.

**Action** Obtain diagnostic information by using the HELP LAST command to display the last message. Additional information may also be obtained from the MANTIS error log file.

**218** `A parameter is incorrectly specified.`

**Explanation** A parameter that is not allowed in a GET, UPDATE, INSERT, or DELETE statement has been specified, or a keyword is incorrectly spelled, or the parameters are out of order, or the value of the level-number parameter is greater than the number of *levels*.

**Action** Correct the syntax of the statement or the spelling of the parameters.

**219** `The symbolic name was not defined with a levels parameter.`

**Explanation** You can specify a LEVEL=*level-number* parameter only if the symbolic name was defined by an ACCESS or FILE statement with a *levels* parameter.

**Action** Decide whether to use a single level or multiple levels.

**220** `A key parameter is incorrectly specified.`

**Explanation** The key parameter on a GET statement should consist of one or more expressions separated by commas and enclosed in parentheses. MANTIS assigns each key value to the corresponding key element in the file profile or user view, so the types and sizes must match. The number of key values must not exceed the number of key elements. For a MANTIS file or external file, you can specify a partial key with fewer key values than key elements.

**Action** Correct the key parameter.

**221** `Record deletion failed.`

**Explanation** MANTIS received an error status when attempting to delete a record from a MANTIS file or external file.

**Action** Obtain diagnostic information by using the HELP LAST command to display the last message. Additional information may also be obtained from the MANTIS error log file.

**222** `The requested operation is not allowed for this type of file.`

**Explanation** An attempt was made to read a record from a sequential file using GET...(key,...), or to update or delete a record in a sequential file.

**Action** Rework your program to use an allowed operation or a different type of file.

**223** `Record update failed.`

**Explanation** MANTIS received an error status when attempting to update a record in a MANTIS file or external file.

**Action** Obtain diagnostic information by using the HELP LAST command to display the last message.  Additional information may also be obtained from the MANTIS error log file.

**249** `The symbolic name was not defined by a SCREEN statement.`

**Explanation** A CONVERSE keyword must be followed by the symbolic name of a screen defined by a SCREEN statement.

**Action** Supply the correct symbolic name.

**250** `The record retrieved contains an invalid numeric field.`

**Explanation** The record contains a numeric field that cannot be converted to a BIG or SMALL numeric value.

**Action** Check whether the file profile is correct.

**251** `A MANTIS variable is the wrong type for conversion to external format.`

**Explanation** The type of the MANTIS variable is incompatible with the external format specified in the profile.

**Action** Identify and correct the source of the conversion error.

**252** `An error occurred converting MANTIS data to external format.`

**Explanation** A conversion error has occurred while attempting to convert MANTIS data to an external data format. This may be due to overflow or bad data.

**Action** Identify and correct the source of the conversion error.

**253** `A MANTIS variable is the wrong type for conversion to internal format.`

**Explanation** The type of the external variable specified in the profile is incompatible with the internal MANTIS format.

**Action** Identify and correct the source of the conversion error.

**254** `An error occurred converting external data to MANTIS internal format.`

**Explanation** A conversion error has occurred while attempting to convert data from an external data format to internal MANTIS format. This may be due to bad data or overflow.

**Action** Identify and correct the source of the conversion error.

**260** `The input line is too long.`

**Explanation** While reading a Keyboard File, MANTIS encountered a line that exceeded the maximum line length of 256 characters.

**Action** Edit the Keyboard File to break the line into multiple lines.

**261** `An attempt to open a print file has failed.`

**Explanation** The print file specified in a PRINTER statement (or the default print file specified in your user profile) cannot be opened.

**Action** Use HELP LAST to identify the cause of the failure and enter a correct PRINTER statement. Additional information may also be obtained from the MANTIS error log file.

**262** `An attempt to write to the print file has failed.`

**Explanation** MANTIS cannot write to the print file specified in a PRINTER statement (or the default print file specified in your user profile).

**Action** Use HELP LAST to identify the cause of the failure and enter a correct PRINTER statement. Additional information may also be obtained from the MANTIS error log file.

**263** `An attempt to close the print file has failed.`

**Explanation** MANTIS cannot close the print file that you have specified in a PRINTER statement (or the default print file specified in your user profile).

**Action** Use HELP LAST to identify the cause of the failure and enter a correct PRINTER statement. Additional information may also be obtained from the MANTIS error log file.

**264** `The CLASS name is invalid.`

**Explanation** The CLASS name that was specified in either an ATTRIBUTE(TERMINAL) or ATTRIBUTE(PRINTER) statement is invalid.

**Action** For the ATTRIBUTE(TERMINAL) statement, verify that the CLASS name is one of the reserved names for the type of screen output being generated. For the ATTRIBUTE(PRINTER) statement, verify that the CLASS name has been defined through the Update Printer Definitions option of the Master User Facility.

**265** `The CLASS of an open device cannot be changed.`

**Explanation** The ATTRIBUTE(TERMINAL) statement cannot be used to change the TERMINAL CLASS. The ATTRIBUTE(PRINTER) statement cannot be used to change the PRINTER CLASS while output is directed to the PRINTER.

**Action** For the ATTRIBUTE(TERMINAL) statement, specify the desired TERMINAL CLASS through the configuration file. For the ATTRIBUTE(PRINTER) statement, close the PRINTER using the OUTPUT SCREEN statement before attempting to change the PRINTER CLASS.

**281** `The template parameter is not a TEXT variable.`

**Explanation** The optional template parameter in a CONVERSE statement must be a TEXT variable.

**Action** Define the template parameter as a TEXT variable.

**282** `The template parameter is not supported in compatibility mode.`

**Explanation** The optional template parameter in a CONVERSE statement cannot be used in compatibility mode because it is not supported by MANTIS for the IBM mainframe.

**Action** Do not use the template parameter in compatibility mode or use the Update Configuration File option in the Master User Facility to change the compatibility mode setting.

**301** `Your program is not set up correctly.`

**Explanation** This is an internal error.

**Action** Contact your Cincom representative.

**302** `There is nowhere to return to from the EXIT or RETURN statement.`

**Explanation** The current subprogram was not called by a DO statement, as expected. That is, there is no known caller of this subprogram able to continue program execution.

**Action** A subprogram cannot be RUN, it must be called by a DO statement. Typing “RUN” <start line>“, where the start line is part of a subprogram, generates this fault.

**303** `Potential program loop encountered. Enter ‘KILL’ to terminate.`

**Explanation** Your program has executed more statements than the SLOT and SLICE values allow. It may be in an endless loop.

**Action** Press ENTER to continue execution of the program. Press KILL or enter KILL in the Key Simulation field to terminate execution. If you intended your program to execute this many statements without interaction from the terminal, increase the SLOT or SLICE values to avoid this warning message.

**304** `MANTIS is unable to acquire any more memory.`

**Explanation** MANTIS is unable to acquire more memory from the operating system. All internally held memory has been released and MANTIS is still unable to acquire memory.

**Action** Provide MANTIS with more memory or check for possible user errors: recursive external subprogram calls (check DOLEVEL), too many large external subprograms active, declaration of large arrays and strings (check dimensions).

**305** `MANTIS is unable to acquire the requested memory.`

**Explanation** MANTIS is unable to perform a user processing request because the request would require too much memory. MANTIS makes no attempt to perform the processing requested.

**Action** Examine the statement that allocates the array to make sure that dimension variables are within the permitted limits.

**310** `Error in Function argument (##......##).`

**Explanation** The argument to a built-in mathematical function (e.g., SIN, EXP) is invalid. The exact cause will be shown.

**Action** Check your program logic to avoid the arithmetic exception condition.

**311** `Floating Point Error (##......##).`

**Explanation** A Floating Point Processor Exception occurred while evaluating a numeric expression. The exact cause will be shown.

**Action** Check your program logic to avoid the arithmetic exception condition.

**320** `The program has been stopped by the user.`

**Explanation** You have pressed CTRL-BREAK, which stops program execution.

**Action** You are now back in either MANTIS programming mode or your Facility Selection menu. Enter the command, statement, or option of your choice.

**330** `An error occurred when trying to perform #####.`

**Explanation** An error occurred when MANTIS attempted to perform a DOS command.

**Action** Use HELP LAST to identify the error and try again.

**350** `The file containing the list of resident programs cannot be found.`

**Explanation** The file containing the list of resident programs could not be found when attempting to load the resident programs during MANTIS initialization. MANTIS will continue without loading resident programs.

**Action** Inform your Cincom representative of this internal error.

**351** `MANTIS cannot allocate memory for the list of resident programs.`

**Explanation** Memory cannot be allocated for the list of resident programs. MANTIS will continue without using resident programs.

**Action** Provide MANTIS with more memory or reduce the number of resident programs.

**352** `The resident program # cannot be found.`

**Explanation** A program referenced in the list of resident programs is not in the MANTIS file, or the program has been bound and the source deleted, and you are running a version of MANTIS that does not support bound programs. MANTIS will continue without including the program in the resident program pool.

**Action** Put a copy of the program in the MANTIS file or remove/correct the name in the resident program list.

**353** `The resident program # contains logic errors.`

**Explanation** A program referenced in the list of resident programs cannot be used if it was saved with logic errors (“unresolved logic”).  MANTIS will continue processing without including the program with the resident programs.

**Action** The program concerned is specified in the error message.  Correct the program logic in that program. The program logic can be checked by either the BIND or RUN commands.

**354** `MANTIS cannot allocate memory for the resident programs.`

**Explanation** Memory cannot be allocated for some or all of the resident programs.  MANTIS will continue processing without including these programs with the resident programs.

**Action** Provide MANTIS with more memory.

**355** `The resident program # cannot be loaded.`

**Explanation** MANTIS failed to load the specified resident program.

**Action** Examine the MANTIS error log file for additional information on the specific reason for the failure, and take appropriate action.

**400** `A symbolic name is missing.`

**Explanation** A symbolic name does not immediately follow a statement keyword that requires one. The missing symbolic name could be one of the following:

- A big, small, or text *variable-name* following BIG, SMALL, or TEXT
- A *screen-name* following SCREEN or CONVERSE
- A screen-name in parentheses following ATTRIBUTE
- A *file-name* following FILE or TRAP
- An *access-name* following ACCESS or TRAP
- A *program-name* following PROGRAM

**Action** Insert a symbolic name of the appropriate type.

**401** `The specified string length exceeds the allowed limit.`

**Explanation** The maximum string length specified in a text variable definition is greater than the maximum length set in the configuration file.

**Action** Modify your program logic to use smaller strings, or use the Update Configuration File option in the Master User Facility to increase the maximum string length.

**402** `Too many dimensions have been specified.`

**Explanation** Too many dimensions were specified in an array declaration. The number of dimensions exceeds the maximum number of dimensions set through the configuration file.

**Action** Modify your program logic to use fewer dimensions, or use the Update Configuration File option in the Master User Facility to increase the maximum number of dimensions.

**403** `An array dimension is out of range.`

**Explanation** A dimension specified in an array declaration exceeds the maximum dimension size set through the configuration file.

**Action** Modify your program logic to use smaller dimensions, or use the Update Configuration File option in the Master User Facility to increase the maximum dimension size.

**404** `A dimension list is invalid or incorrectly terminated.`

**Explanation** A dimension specified in an array declaration is not an arithmetic expression, or the list of dimensions is improperly terminated by a right parenthesis.

**Action** Correct the dimension list.

**405** `A symbolic variable reference is incorrectly specified.`

**Explanation** A symbolic variable reference is a numeric or string variable optionally followed by a list of subscripts separated by commas within parentheses.  The subscripts may be array subscripts and/or substring positions.

**Action** Check the statement syntax, that the variable is of the correct data type, and that any subscripts are valid numeric expressions.

**406** `An assignment operator is missing.`

**Explanation** The assignment operator is expected in the following circumstances:

- After the symbolic name following a LET keyword. The LET keyword is optional in MANTIS statements so that if a symbolic name is encountered first on a statement line, an assignment operator is expected. Note that PRINTER is like a symbolic name in this case and must be followed by the assignment operator. Also note that a list of array subscripts in parentheses or the ROUNDED keyword can precede the assignment operator.
- After the closing parenthesis in the ATTRIBUTE statement and before the list of screen attributes that is expected to follow the assignment operator.

**Action** Correct the syntax of the statement.

**407** `Rounding is valid only for a numeric variable.`

**Explanation** Only a BIG or SMALL variable or array element can be qualified by the ROUNDED keyword on the left side of an assignment operator.

**Action** Use ROUNDED only with numeric variables.

**408** `The receiving variable type is incompatible with an assigned expression.`

**Explanation** The receiving variable or array in a LET statement must be compatible in type with the expression(s) on the right side of the assignment operator. Only arithmetic expressions can be assigned to BIG and SMALL variables. Only text expressions can be assigned to TEXT variables.

**Action** Change the type of the variable or change the type of the expression.

**409** `Too many expressions are assigned to the receiving variable.`

**Explanation** A list of expressions separated by commas can be assigned to consecutive elements of an array, given the starting array element on the left side of the assignment operator. The number of expressions allowed is determined by the number of array elements present after the starting array element. Only one expression can be assigned to a variable that is not an array.

**Action** Reconcile the number of expressions with the dimensions or starting element of the receiving variable.

**410** `An end-of-statement marker is missing.`

**Explanation** MANTIS has identified all of the parameters of a statement or command according to the syntax, without reaching a valid end-of-statement marker (a colon or end-of-line).

**Action** Add the end-of-line marker.

**411** `The command or statement keyword is invalid.`

**Explanation** A command or statement must begin with a valid command or statement keyword.

**Action** Enter HELP RESERVED to display a list of valid command and statement keywords.

**413** `The parameter must be an arithmetic expression.`

**Explanation** This message is generated by the UP and DOWN commands, and by the SLICE and SLOT statements, that all require a numeric parameter following the keyword.

**Action** Use an expression with a numeric value in these statements.

**414** `Too many arguments have been specified.`

**Explanation** The maximum number of arguments allowed in an ENTRY statement is 255, and the maximum number of parameters allowed in a CHAIN statement is 40.

**Action** Reduce the number of arguments or parameters.

**415** `A libname parameter is missing.`

**Explanation** In the CHAIN and PROMPT statements, the parameter following the statement keyword must be a text expression that evaluates to a library name.

**Action** Supply a *libname* parameter.

**416** `The program line number is invalid.`

**Explanation** The RUN command has two optional parameters: a *libname* and a program line number. If the first parameter specified is not a text *libname* (expressions are not allowed) then it is assumed to be a program line number and must be an arithmetic expression.

**Action** Specify the *libname* or the line number in the correct format.

**417** `The first parameter must be a libname or a program line number.`

**Explanation** The first parameter of a RUN or COPY command does not satisfy the syntax of the command. The parameter in error is neither a text nor an arithmetic expression and is not an expected keyword (FIRST, LAST or AFTER).

**Action** Correct the syntax of the command or the spelling of the parameters.

**418** `The supplied starting line number is invalid.`

**Explanation** In the ALTER, CHANGE, COPY, ERASE, LIST, and USAGE commands, the parameter specifying the starting program line number must be an arithmetic expression.

**Action** Correct the line number parameter.

**419** `The expression is the wrong type for a logic statement.`

**Explanation** The IF, UNTIL, WHEN, and WHILE statements require a relational expression or an arithmetic expression after the keyword. Both types of expression have numeric values. Nonzero values represent TRUE; zero values represent FALSE.

**Action** Use a relational expression or an arithmetic expression in the logic statement.

**420** `The ending line number was not found in the program.`

**Explanation** In the COPY and ERASE commands, the parameter specifying the ending program line number must evaluate to an existing line number in your program. These are the only commands that require an exact ending program line number.

**Action** Specify the ending line number correctly.

**421** `The starting line number was not found in the program.`

**Explanation** In the ALTER, CHANGE, COPY, ERASE, and RUN commands, the parameter specifying the starting program line number must evaluate to a line number in your program. The above commands require that you specify the exact starting line number. In the LIST and USAGE commands, the specified starting line number must be within the program limits.

**Action** Specify the starting line number correctly.

**422** `The supplied ending line number is invalid.`

**Explanation** In the ALTER, COPY, and ERASE commands, the parameter for the ending program line number must be an arithmetic expression.

**Action** Specify the ending line number correctly.

**423** `An arithmetic expression is required for the line increment.`

**Explanation** The program line increment parameter for the SEQUENCE command must be an arithmetic expression.

**Action** Specify the line increment parameter correctly.

**424** `The starting line number overlaps previous program lines.`

**Explanation** When the SEQUENCE command is used to sequence a subprogram, the starting line number specified must be greater than the program line number immediately preceding the ENTRY statement of the specified subprogram.

**Action** Specify the starting line number correctly.

**425** `The first parameter must be an entry-name or a line number.`

**Explanation** The first parameter of the SEQUENCE statement must be either a subprogram *entry-name* (if you want to sequence only that subprogram) or a program line number. An *entry-name* or an arithmetic expression is expected.

**Action** Correctly specify the *entry-name* or line number.

**426** `The specified subprogram was not found.`

**Explanation** The *entry-name* supplied in the SEQUENCE command was not found in the current program.

**Action** Check that the symbolic name you supplied is a valid *entry-name* by entering the command: DISPLAY *entry-name*.

**428** `You are not authorized to use this statement.`

**Explanation** This statement is reserved for Cincom use only.

**Action** Do not use this statement.

**429** `The parameter must be a text expression.`

**Explanation** The ENQUEUE and PRINTER statements require a text expression as a parameter.

**Action** Specify the resource or device parameter as a text expression.

**430** `The symbolic name was not defined in a FILE or ACCESS statement.`

**Explanation** The TRAP statement requires the symbolic name of a MANTIS file or external file. Such a symbolic name must be defined by a FILE or ACCESS statement.

**Action** Specify the correct symbolic name.

**431** `The windowing row and/or column increments are invalid.`

**Explanation** The SCROLL statement accepts row and column windowing increments. They must be numeric expressions separated by a comma. For example: SCROLL 10,20 sets windowing increments of 10 rows and 20 columns. This is the number of rows scrolled when the WINUP/WINDOWN key is pressed and the number of columns scrolled when the WINLEFT/WINRIGHT key is pressed. The keywords ON and OFF are also valid but have no effect; they are supported for compatibility with MANTIS for the IBM mainframe only.

**Action** Correct the row or column increments.

**432** `You are not allowed access to this program.`

**Explanation** If you do not own a program that you loaded or executed during Program Design, you may not execute any commands while the program is in your work area, except UP, DOWN, and QUIT. Additionally, in order for you to copy program lines from another user’s library, your current program password must match that of the specified source program.

**Action** Dot not try to access the program, or change your program’s password.

**434** `The number of lines is not an arithmetic expression.`

**Explanation** The final optional parameter in a CHANGE, LIST, or USAGE command specifies the number of program lines that the command is to process. This parameter must be an arithmetic expression.

**Action** Specify the number of lines correctly.

**435** `An ENTRY statement was encountered during program execution.`

**Explanation** You must include a STOP statement at the end of your main program if it is followed by any internal subprograms.

**Action** Add the missing STOP statement.

**437** `A symbolic name is missing for the entry-name.`

**Explanation** The program name is not specified correctly in the ENTRY statement of the called program.

**Action** Correct the *entry-name* parameter.

**438** `The symbolic name specified is not a valid program-name or entry-name.`

**Explanation** The symbolic name specified in the DO statement is not the name of a program defined by a PROGRAM or ENTRY statement.

**Action** Specify the name of the subprogram correctly.

**439** `The limit of nested external subprogram calls has been reached.`

**Explanation** A nested external subprogram is one that was itself called from an active external subprogram. The limit of such nested calls is set through the configuration file.

**Action** Modify your program logic to use fewer DOLEVELS, or use the Update Configuration File option in the Master User Facility to increase the maximum number of external DOLEVELS.

**440** `There is no ENTRY statement on the called subprogram.`

**Explanation** A program invoked by external DO or CHAIN must have an ENTRY statement as the first statement.

**Action** Ensure that the program being called is enclosed within an ENTRY-EXIT pair, and that the ENTRY is the first statement.

| | | |
|---|---|---|
| **441** | `The number of actual parameters is incorrect.` | |
| | **Explanation** | The number of arguments on a DO or CHAIN statement must match the number of formal parameters specified on the ENTRY statement of the called program. Otherwise, if fewer arguments than necessary are specified, MANTIS supplies extras and sets them to zero. |
| | **Action** | Correct the number of arguments/parameters specified, either on the DO or the ENTRY statement. |
| **442** | `You have attempted to recursively DO an internal subprogram.` | |
| | **Explanation** | A subprogram may not call itself using internal DO. |
| | **Action** | Change the internal subprogram to a self-contained program and use external DO to call it. From within the subprogram, also use external DO to call the program itself. |
| **443** | `An ENTRY statement parameter is not a MANTIS variable name.` | |
| | **Explanation** | The formal parameters on an ENTRY statement must be MANTIS variable names; expressions and reserved words are not allowed as parameters. |
| | **Action** | Correct the parameter on the ENTRY statement. |
| **445** | `An argument is specified as an expression in compatibility mode.` | |
| | **Explanation** | When running in compatibility mode, arguments on a CHAIN or DO statement must be MANTIS variable names; expressions are not permitted. |
| | **Action** | Either assign the desired expression to a suitable MANTIS variable and pass this variable as the argument or use the Update Configuration File option in the Master User Facility to turn off compatibility mode. |
| **446** | `Text literal expected in CHANGE command.` | |
| | **Explanation** | You have incorrectly specified a CHANGE command, possibly by omitting a delimiter (single quote, comma, or semicolon) or a string. |
| | **Action** | Correct the CHANGE command. |

**447** `A terminating delimiter is missing.`

**Explanation** The CHANGE command requires the new and old strings to be delimited by a comma, semicolon, or apostrophe.

**Action** Check that all required delimiters are present and match.

**448** `The supplied password is invalid.`

**Explanation** The password parameter on the PROGRAM statement must be a valid text expression.

**Action** Correct the password parameter.

**449** `The line number specified to copy lines after is invalid.`

**Explanation** The parameter following the keyword AFTER must be an arithmetic expression that evaluates to a valid program line number.

**Action** Correct the line number parameter following the AFTER keyword.

**450** `You are not permitted to access the source program.`

**Explanation** In order for you to COPY program lines from another user's library, your current program password must match that of the specified source program.

**Action** Either use REPLACE to change your program's password to that of the source program or LOAD the desired source program from the other user's library and SAVE it in your own library before doing the COPY.

**451** `The required FIRST, LAST or AFTER destination parameter is missing.`

**Explanation** The COPY command must contain a destination parameter, which is missing. The destination parameter must be one of FIRST, LAST, or AFTER.

**Action** Supply the required destination parameter.

**452** `The specified source program is empty.`

**Explanation** The source program specified in the COPY statement does not contain any program statements. No lines were copied.

**Action** Specify a valid source program.

**453** `The specified destination program is empty.`

**Explanation** The AFTER keyword cannot be used to copy to an empty program.

**Action** Specify FIRST or LAST as the destination position.

**454** `The source (end) parameter must be LAST or a line number.`

**Explanation** The parameter in the COPY statement that defines the end of the source program segment is invalid. The keyword LAST or a valid line number must be specified in that position.

**Action** Specify LAST or a valid line number for that parameter.

**455** `The prompter does not exist.`

**Explanation** The key of the prompter as specified in the PROMPT or HELP statement does not exist in the library.

**Action** Specify a key for an existing prompter.

**456** `All referenced data types must be one of BIG, SMALL or TEXT.`

**Explanation** All data types referenced in all expressions and in some statements must be one of BIG, SMALL, or TEXT. One of the data types in the statement is of another type.

**Action** Use the correct data type for the statement.

**457** `The required ALL or symbolic name parameter is missing.`

**Explanation** The DISPLAY command requires a parameter, which must be a symbolic name or ALL.

**Action** Supply the required DISPLAY parameter.

**458** `The parameter must be ALL or a symbolic name.`

**Explanation** The DISPLAY command requires a parameter, which must be a symbolic name or ALL. Symbolic names are names that you have given to program variables of any data type.

**Action** Specify a valid symbolic name or ALL.

**459** `There is no HELP available on this subject.`

**Explanation** MANTIS cannot find any HELP information that matches the parameter on the HELP command.

**Action** Check that your HELP command parameter is either a valid error message number, a valid name as listed by HELP RESERVED, or LAST.

**460** `The parameter must be a text expression.`

**Explanation** The PERFORM statement requires a parameter that must be a text expression.

**Action** Specify a valid text expression.

**462** `The UPPERCASE function is not supported in compatibility mode.`

**Explanation** The built-in MANTIS text function UPPERCASE is not supported in MANTIS for the IBM mainframe, and compatibility mode is in effect.

**Action** Avoid using the UPPERCASE function or use the Update Configuration File option in the Master User Facility to change the compatibility mode setting.

**463** `The destination parameter must be SCREEN or PRINTER.`

**Explanation** The OUTPUT statement requires a destination parameter that must be SCREEN, PRINTER, or SCREEN PRINTER.

**Action** Supply the required destination parameter.

**464** `A delimiter is invalid or missing.`

**Explanation** The CHANGE and USAGE commands expect the new and old strings to be delimited by a comma, semicolon, or apostrophe.  The first delimiter is invalid or missing.

**Action** Supply the correct delimiter.

**467** `The function specified for CLEAR is not recognized.`

**Explanation** The CLEAR verb accepts one or more of the following parameter values: ALL, TRAP, BREAK, any program variable name (simple or complex, scalar or array).

**Action** Use one of the above parameters.

**468** `The function specified for SET is not recognized.`

**Explanation** The SET verb accepts either BREAK, TRAP, or $SYMBOL as a parameter value.

**Action** Replace the invalid parameter with BREAK, TRAP, or $SYMBOL.

**469** `The screen-name parameter must be a symbolic name.`

**Explanation** The optional screen name parameter in the CLEAR statement must be a MANTIS symbolic name, which has been defined previously in a SCREEN statement.

**Action** Specify a symbolic name.

**470** `The exit parameter must be a text expression.`

**Explanation** The optional printer exit parameter in the OUTPUT statement must be a valid 1–8 character text expression that contains the printer exit program name.  This parameter is supported for compatibility with MANTIS for the IBM mainframe only.

**Action** Specify a valid text expression.

**471** `The symbolic name is not a valid screen name.`

**Explanation** The optional screen name parameter in the CLEAR statement and the required screen name parameter in the ATTRIBUTE statement must be a MANTIS symbolic name that has been previously defined in a SCREEN statement.

**Action** Specify a valid screen name for the symbolic name.

**472** `The field-name parameter must be a symbolic name.`

**Explanation** The optional field name parameter in the ATTRIBUTE command must be a MANTIS symbolic name that is the name of a field in the specified screen.

**Action** Specify a symbolic name.

**473** `The field-name is not defined in the screen.`

**Explanation** The optional field name parameter in the ATTRIBUTE statement must be a MANTIS symbolic name that is the name of a field in the specified screen. The specified field name is not defined, or it has not been defined in the specified screen.

**Action** Specify a field name that is defined in the specified screen.

**474** `The attribute specification is invalid.`

**Explanation** The attribute parameter in the ATTRIBUTE statement must be a valid text expression containing valid attributes, or their 3-character abbreviations, separated by commas.

**Action** See “ATTRIBUTE” on page 81, and change the specification.

**475** `The DISPLAY parameter is invalid.`

**Explanation** A text expression is required.

**Action** Refer to the *MANTIS for Windows Facilities Reference Manual*, P19-2301, for a list of valid DISPLAY command parameters.

**476** `The receiving variable must be a symbolic name.`

**Explanation** The receiving variable parameter in the OBTAIN statement must be a valid MANTIS symbolic name.

**Action** Specify a valid symbolic name.

**478** `The numeric value has caused an overflow or underflow.`

**Explanation** The numeric input received by the OBTAIN statement is invalid because it is too large or too small to be stored internally by MANTIS. This is a hardware floating-point limitation. The maximum range is approximately 1E-308 to 1E308.

**Action** Specify a value within the acceptable range.

**480** `The tab value must be an arithmetic expression.`

**Explanation** The optional AT tab parameter in the SHOW statement must be an arithmetic expression.

**Action** Specify an arithmetic expression.

**481** `The required heading parameter is missing or invalid.`

**Explanation** The parameter in the HEAD statement must be a text expression.

**Action** Supply a valid heading parameter.

**482** `The line number parameter is invalid.`

**Explanation** The line number parameter in the CLEAR BREAK or SET BREAK command is not valid. It must be an arithmetic expression that evaluates to a line number in your program.

**Action** Specify a valid line number.

**483** `A breakpoint is not set for that line number.`

**Explanation** The line number specified in the CLEAR BREAK command did not have a breakpoint set for it.

**Action** Specify a line number for which a breakpoint has been defined.

**484** `The command parameter must be a text expression.`

| | |
|---|---|
| **Explanation** | The optional command parameter in the SET BREAK command must be a valid text expression that evaluates to a MANTIS statement or command. |
| **Action** | Specify a text expression. |

**485** `The program line does not exist.`

| | |
|---|---|
| **Explanation** | The program line parameter in the SET BREAK command must evaluate to a line number that exists in your program. The specified line number does not exist. |
| **Action** | Specify an existing line number. |

**486** `The breakpoint table is full.`

| | |
|---|---|
| **Explanation** | The breakpoint table, which is a finite size, has been filled. No further breakpoints may be set until existing breakpoints are cleared. |
| **Action** | Clear some existing breakpoints to make room for new ones. |

**487** `The program has stopped at a breakpoint.`

| | |
|---|---|
| **Explanation** | A breakpoint was set at the current line number, and control was passed to it. MANTIS temporarily stopped execution of your program at this point, and placed you in programming mode. MANTIS will continue execution of your program from this point when you enter GO. |
| **Action** | Enter the command(s) of your choice, and enter GO when ready to continue. |

**490** `The parameter data type is not valid.`

**Explanation** The parameters in DO and CHAIN statements have the following restrictions: ENTRY, SCREEN, FILE and ACCESS variables are not allowed as CHAIN parameters.

ENTRY variables are not allowed as DO parameters.

**Action** Restructure the application to use external DO instead of CHAIN. Complex variables may be passed (by reference) to external subprograms, but in that case the fields defined in the complex variable cannot be referenced in the external subprogram.

**493** `The parameter must be the symbolic name of a text variable.`

**Explanation** The text parameter in the FSI function must be a valid MANTIS symbolic name that has been defined as a text variable.

**Action** Specify the symbolic name of a text variable.

**494** `The access name must be a symbolic name.`

**Explanation** The *access-name* parameter in the FSI function must be a valid MANTIS symbolic name.

**Action** Specify a valid symbolic name.

**495** `The access does not exist.`

**Explanation** The *access-name* specified in the FSI function does not exist.

**Action** Specify an access name that exists.

**496** `The line number must not define a continuation line.`

**Explanation** The line number parameter in the SET BREAK command defines a continuation line. Breakpoints cannot be set on continuation lines.

**Action** Specify a line number that does not start with an apostrophe.

**497** `An access-name or program-name is missing.`

**Explanation** The RELEASE statement requires either an *access-name* or a *program-name* as previously defined in an ACCESS or PROGRAM declaration.

**Action** Use the DISPLAY command to check the data type of the parameter you have specified.

**498** `The CLEAR parameter is invalid.`

**Explanation** An ENTRY or PROGRAM variable has been specified as a parameter to the CLEAR statement.

**Action** Remove all ENTRY and PROGRAM variables from the CLEAR statement.

**500** `The parameter value must be positive.`

**Explanation** A nonpositive parameter has been specified in the CHANGE, GO, UP/DOWN, LIST, SEQUENCE, or USAGE command where a positive parameter is required.

**Action** Specify a positive parameter.

**501** `Your program is not set up correctly.`

**Explanation** This error can be caused by using a program design command as a statement in your program. For example, “SHOW BREAK” in a program causes this error.

**Action** Check your program for program design commands.

**502** `Your program does not contain any lines.`

**Explanation** You cannot CHAIN to or externally DO a null program.

**Action** Make sure that your program contains at least one line.

**503** `The libname parameter is either missing or incorrectly specified.`

**Explanation** The SAVE, LOAD, and PURGE commands all require that a *libname* be specified as the first parameter. This parameter is optional on REPLACE. The *libname* parameter specifies the name of an entity in a user's library in the format [*user name*:]*entity-name*. The user name cannot exceed 16 characters. The *entity-name* cannot exceed 30 characters.

**Action** Either supply or correct the *libname* parameter.

**504** `The password parameter is incorrectly specified.`

**Explanation** The password parameter must be a text expression that evaluates to a value of not more than 16 characters. This parameter is optional on the LOAD, SAVE, REPLACE, and PURGE commands.

**Action** Correct the password parameter.

**505** `The description parameter is not a valid text expression.`

**Explanation** The description parameter must be a text expression. It may be specified as any length but is always truncated to 48 characters.

**Action** Correct the description parameter.

**506** `The specified program already exists.`

**Explanation** The SAVE command can be used only to save a program that does not already exist.

**Action** Use the REPLACE command to replace an existing program.

**507** `An unexpected parameter is specified.`

**Explanation** After processing all parameters on a LOAD, SAVE, REPLACE, PURGE, NEW, or EDIT command, another unexpected parameter was found.

**Action** Remove the unexpected parameter.

**508** `You are not permitted the requested access to another user's library.`

**Explanation** No user can access entities in the CONTROL user's library. All users can access entities and LOAD programs from all other users' libraries, provided the correct password is supplied. All users can execute any program belonging to another user, but they cannot edit it unless the correct password is supplied. Only the Master User can SAVE, REPLACE, and PURGE programs in another user's library.

**Action** Either supply the correct password, or ask a user with sufficient authority to perform the desired operation.

**509** `The program does not exist.`

**Explanation** There is no program with the specified name in the library. You can use the Directory Facility to list the programs in your own library. To search another user's library, you must specify the program name following a user name and colon.

**Action** Correct the program name parameter.

**511** `The supplied password is incorrect.`

**Explanation** To LOAD a program from another user's library or to PURGE a program from your own library, you must supply the correct password, either explicitly or by default. The default password is either the password of the most recently LOADed program or your sign-on password if this is the first LOAD command issued since entering Program Design.

**Action** Supply the correct password.

**512** `Program deletion failed.`

**Explanation** MANTIS received an error status when attempting to delete a program from the MANTIS file.

**Action** Obtain diagnostic information by using the HELP LAST command to display the last message.

**513** `Program replace failed.`

**Explanation** MANTIS received an error status when attempting to update a program on the MANTIS file.

**Action** Obtain diagnostic information by using the HELP LAST command to display the last message.

**515** `The file is reserved, no access is permitted.`

**Explanation** The FILE statement specifies a file name that is one of the files reserved for MANTIS internal use only.

**Action** You may not access the specified file.

**516** `The program is too large to save.`

**Explanation** The program you are trying to save exceeds the permitted limits.  These limits depend on the size of the program in lines, the number and size of the user words in the program, and the size and number of debugging commands.

**Action** Save your program to a file using the SAVE FILE command.  Then use your external text editor to break your program into several smaller programs.

**517** `Overflow in line ##### during CHANGE.`

**Explanation** You have attempted to change a line so that it exceeds the maximum permitted line length of 255 characters.

**Action** Modify your CHANGE statement, or break up your line before you try to change it.

**520** `The DOS file specification is missing or invalid.`

**Explanation** A DOS file specification must follow the FILE keyword on the EDIT, SAVE, LOAD, and REPLACE commands.  It must be a valid DOS file specification.

**Action** Correct the DOS file specification parameter.

**523** `Program load from a DOS file failed.`

**Explanation** MANTIS received an error status when reading a MANTIS program from a file on a LOAD FILE command or upon returning from an EDIT FILE command.

**Action** Obtain diagnostic information by using the HELP LAST command to display the last message.

**524** `Program save to a DOS file failed.`

**Explanation** MANTIS received an error status when writing a MANTIS program to a file on a SAVE FILE, REPLACE FILE, or an EDIT FILE command.

**Action** Obtain diagnostic information by using the HELP LAST command to display the last message.

**528** `Cannot EDIT - no editor name available.`

**Explanation** You have issued an EDIT command, but no editor name is available to MANTIS.

**Action** Provide the editor name by specifying it in your configuration file, or by specifying it in your environment (with the MANTIS SET $SYMBOL statement).

**529** `Could not create a temporary file.`

**Explanation** MANTIS received an error status while trying to create a temporary file for the external editor.

**Action** Use HELP LAST to examine the error message.

**530** `LIST EDIT linecount is invalid.`

**Explanation** This is an internal error.

**Action** Contact your Cincom representative.

**588** `DOS file COMMIT failed.`

**Explanation** This is an internal error.

**Action** Contact your Cincom representative.

**589** `DOS file RESET failed.`

**Explanation** This is an internal error.

**Action** Contact your Cincom representative.

**594** `The COMMIT ON/OFF statement is not supported in compatibility mode.`

**Explanation** The ON and OFF keywords are not allowed in the COMMIT statement in the IBM mainframe version of MANTIS.

**Action** You must use the Update Configuration File option in the Master User Facility to turn off compatibility mode before you can use the ON/OFF keywords in a program COMMIT statement.

**596** `The NUMERIC function is not supported in compatibility mode.`

**Explanation** The NUMERIC function is not yet available in MANTIS for the IBM mainframe.

**Action** You must use the Update Configuration File option in the Master User Facility to turn off compatibility mode before you can use this function in a program.

**597** `The NUMERIC function requires a TEXT argument.`

**Explanation** A TEXT variable or literal must be supplied as the only argument to the NUMERIC function.

**Action** Replace the function argument with a TEXT value.

**598** `The SQR function requires a positive numeric argument.`

**Explanation** The argument to the SQR function must evaluate to a positive numeric value.

**Action** Replace the function argument.

**599** `This function is not available.`

**Explanation** The requested function or facility is an optional feature that is not supplied in your configuration of MANTIS.

**Action** If you wish to use this feature, please contact your Cincom representative and ask for the details.

**600** `An entry-name is defined more than once in the program.`

**Explanation** An *entry-name* is associated with more than one ENTRY statement in the program.

**Action** Use the USAGE command to find all occurrences of ENTRY, and examine each one for a duplicate name.

**602** `Fault Traps are not supported in compatibility mode.`

**Explanation** SET TRAP is incompatible with MANTIS for the IBM mainframe.

**Action** Compatibility mode must be turned off before this feature can be used. Use the Update Configuration File option in the Master User Facility to change the compatibility mode setting.

**603** `A program-name or entry-name must be supplied.`

**Explanation** The optional parameter for SET TRAP is the name of an internal or external subprogram that handles the error condition. This name must be declared either in an ENTRY or PROGRAM statement within the current program.

**Action** Supply the name of the error-handling program in an ENTRY or PROGRAM statement.

**605** `The number of lines to execute is incorrectly supplied.`

**Explanation** The GO command requires a numeric expression giving the number of program statements to execute before a “Program Breakpoint” stop is simulated. The parameter given is not a valid expression giving a numeric result.

**Action** Correct or add a valid expression that evaluates to the number of program statements to be executed.

**606** `The EDIT LIST parameters must be unsubscripted string variables.`

**Explanation** The EDIT LIST statement takes only string variables as parameters. These may be scalar, one-dimensional TEXT, KANJI, or MIXED variables, but array elements may not be specified.

**Action** Use the DISPLAY command to check that the parameter names specified are defined and are one of the text data types.

**607** `No more than one dimension is allowed for EDIT LIST.`

**Explanation** The EDIT LIST statement handles one-dimensional lists only.

**Action** Use the DISPLAY command to check that the parameter names specified are defined as TEXT with no more than one dimension.

**608** `The specified number of program lines has been executed.`

**Explanation** The number of lines specified in the previous GO command was executed. Program execution is suspended at the line displayed.

**Action** Use the GO command to proceed with program execution.

**609** `The function is not recognized ($).`

**Explanation** The function identified by the dollar sign is not recognized. The only valid use of the dollar sign ($) is in the keyword $SYMBOL and as a shorthand form of the PERFORM command.

**Action** Remove the dollar sign.

**610** `The verb is not supported in compatibility mode.`

**Explanation** The following program verbs are not available in MANTIS for the IBM mainframe: BREAK, FOR, NEXT, RETURN. Compatibility mode must be turned off before these verbs can be used.

**Action** Use your Master User facilities to change the compatibility mode setting, or do not use these verbs if you want your program to be compatible with MANTIS for the IBM mainframe.

**611** `The parameters for SIZE have conflicting attributes.`

**Explanation** The two parameters supplied in the SIZE function conflict with each other. The first parameter can be the name of any text or numeric program variable. The second parameter is meaningful as follows: “MAX” applies only to text variable names, “DIM” applies only to array variables, a numeric expression applies only to array variables.

**Action** Correct the parameter in conflict.

**612** `The separator, TO, is missing.`

**Explanation** The FOR statement requires a TO clause to give the final value of the loop counter.

**Action** Look for a syntax error in the FOR clause. Check that the expression assigning an initial value to the loop counter is valid.

**613** `The FOR loop counter variable is invalid.`

**Explanation** The target of the FOR statement assignment must be a numeric variable.

**Action** Use the DISPLAY statement to check the type of the variable.

**615** `The first FORMAT parameter must be a numeric expression.`

**Explanation** The first parameter to the FORMAT function must be a numeric expression that evaluates to the number to be converted to text, according to the edit mask given by the second parameter.

**Action** Check the syntax of the first parameter and verify that it generates a numeric result.

**616** `The second FORMAT parameter must be a text expression.`

**Explanation** The second parameter to the FORMAT function must be a text expression that evaluates to an edit mask that describes the format of the converted number, as given by the first parameter.

**Action** Check the syntax of the second parameter and verify that it generates a text result.

**617** `The optional SIZE parameter is inappropriate or invalid.`

**Explanation** The optional second parameter to the SIZE function must be “DIM”, “MAX”, “BYTE”, or an arithmetic expression that evaluates to a positive array dimension value.  A second parameter is inappropriate if the first parameter is not a defined variable.  An error can occur if you specify one of the following: “MAX” for a numeric array, a text value other than “MAX”, “DIM”, or “BYTE”, a numeric value less than or equal to zero.

**Action** Use the DISPLAY command to examine the individual parameters.  Verify that the parameter contains a valid value.

**619** `A CLEAR parameter is not supported in compatibility mode.`

**Explanation** In MANTIS for the IBM mainframe, the CLEAR verb accepts only a *screen-name* as a parameter.

**Action** Compatibility mode must be turned off before the CLEAR parameter can be used.  Use the Update Configuration File option in the Master User Facility to change the compatibility mode setting.

**620** `This form of the statement is reserved for privileged users.`

**Explanation** Some forms of the ATTRIBUTE statement are only available to the Master User. These are references to TERMINAL or to fields by their logical screen coordinates (*row*, *col*). The ATTRIBUTE function may be used to obtain the attributes of the PRINTER only, (e.g., SHOW ATTRIBUTE(PRINTER)).

**Action** Correct the ATTRIBUTE statement.

**621** `The first PAD/UNPAD parameter must be a string variable (element).`

**Explanation** The first parameter after the PAD or UNPAD keyword is not a TEXT or KANJI variable or array element.

**Action** Change or add a valid parameter.

**622** `The pad character is incorrectly specified.`

**Explanation** The pad character, if specified, must be the second parameter in a PAD or UNPAD statement. This second parameter can be an expression. It must evaluate to the same type (TEXT or KANJI) as the type of variable being padded.

**Action** Ensure that the first PAD or UNPAD parameter and the following expression for the pad character both evaluate to the correct type.

**623** `Cannot create indexed file using secondary key.`

**Explanation** You have specified REPLACE in your ACCESS statement, but the specified view is for an alternate index (key of reference is nonzero).

**Action** Use the NEW parameter in the ACCESS statement if you are attempting to create a secondary index, or use a view defining the primary key (key of reference is zero/omitted) if you want to create a new version of the file.

**624** `Substring subscripts are not allowed with a BEFORE/AFTER/ALL qualifier.`

**Explanation** The qualifiers BEFORE, AFTER, and ALL are not allowed in conjunction with a substring reference in the first PAD/UNPAD parameter. These qualifiers specify where padding is to occur, whereas a substring reference requests an overlaying function and not a padding function. You cannot disjoin a substring from its parent string. The above qualifiers apply to whole strings only.

**Action** Remove the substring subscript or the qualifier as appropriate.

**625** `A text expression is required for the symbol name parameter.`

**Explanation** The SET $SYMBOL statement or command requires a text expression for the symbol name.

**Action** Include the text expression for the symbol name.

**626** `An error occurred assigning the value to the symbol name.`

**Explanation** The SET $SYMBOL operation failed. The error was detected upon return from the system call involved. An error log message should have been generated describing the condition.

**Action** Check the symbol name expression for validity.

**627** `A SHOW parameter is not supported in compatibility mode.`

**Explanation** The BREAK, NEXT, and TRAP keywords are not yet supported in the SHOW statement in the IBM version of MANTIS.

**Action** You must switch off IBM compatibility mode in order to include any of the above SHOW statement variations in a program.

**628** `Invalid indirect variable reference.`

**Explanation** The indirect reference operator (ampersand character) must be followed by either the name of a scalar text variable that contains the symbolic name of a MANTIS variable, or a text expression that evaluates to the symbolic name of a MANTIS variable.

**Action** Ensure that the variable or expression following the ampersand (&) contains a valid symbolic name. The ampersand operator has higher precedence than all other operators except for parentheses around an expression. Therefore, you must always enclose any indirect reference expressions within parentheses. Note that this also applies to array and subscript operators. To use either a member of a text array or a text substring as an indirect reference, it must be enclosed within parentheses. For example, use &(ABC(3)) to indirectly reference via the third element of the array ABC.

**631** `Too many components in key.`

**Explanation** The external file view definition has more than 128 elements with the KEY attribute.

**Action** Use fewer elements for the key.

**632** `The ENQUEUE parameter is not supported in compatibility mode.`

**Explanation** MANTIS for the IBM mainframe does not support the ENQUEUE option for external (ACCESS) files or internal (FILE) files.

**Action** Remove the ENQUEUE parameter or use the Update Configuration File option in the Master User Facility to change the compatibility mode setting.

**633** `The use of NEXT on a keyed GET is not supported in compatibility mode.`

**Explanation** The NEXT keyword on a keyed GET is not supported by MANTIS for the IBM mainframe.

**Action** Do not use the NEXT keyword, or use the Update Configuration File option in the Master User Facility to change the compatibility mode setting.

**634** `The use of KANJI features requires a KANJI version of MANTIS.`

**Explanation** KANJI is not supported in this version of MANTIS for Windows.  A KANJI version of MANTIS is available and is required to properly handle any double-byte character data.

**Action** If you think the version of MANTIS you are running is a KANJI version, and you get this error, contact your Cincom representative.

**635** `A SEQUENTIAL file must have a reference variable in order to do updates.`

**Explanation** In order to update a sequential external DOS file, MANTIS requires a reference variable.  The RBA (Relative Byte Address) of the most recently retrieved record is stored in the reference variable.  This RBA is used if you subsequently UPDATE the record.

**Action** Define a reference variable for this file using the External File Design Facility.

**638** `The external MIXED field has been truncated on input.`

**Explanation** This error occurs when the external mixed field expands on input due to the insertion of internal control codes by MANTIS to separate the ASCII from the KANJI character data.  The mixed field is qualified as containing optional space separators (BLNK).

**Action** You must allow for expansion of the mixed data by leaving some trailing spaces in the external record layout.  Alternatively, ensure that the external data has a space between any adjacent ASCII and KANJI characters, or use the coded external mixed data representation (CODE).

**639** `The screen field coordinates are invalid or do not identify a field.`

**Explanation** When identifying a screen field by its logical coordinates, the specified row and column coordinates must locate a character position within the desired field. Coordinates range from (1,1) to (32767,32767), identifying every character position in the logical display. Note that the relative position of the field in its screen-design is not the same as its position in the logical display, unless the screen-design is CONVERSED at (1,1) in the logical display. This is the default position for any CONVERSE.

**Action** Correct the screen field coordinates.

**641** `Keyboard file input is invalid.`

**Explanation** The format of the current line in the keyboard file is invalid and is rejected.

**Action** Check and fix the invalid line indicated in the accompanying message.

**642** `The screen contains an inconsistent row or column value.`

**Explanation** This is an internal error.

**Action** Contact your Cincom representative.

**643** `The logical display limit has been exceeded.`

**Explanation** The logical display is limited to 32767 rows by 32767 columns.

**Action** Rework the size of the display to comply with the maximum limits.

**644** `An attempt to create the save file has failed.`

**Explanation** The output save file specified in a SAVE FILE command cannot be created.

**Action** Use HELP LAST to identify the cause of the failure and enter a correct SAVE FILE command.

**645** `A numeric expression is required for each CHR parameter.`

**Explanation** The CHR function parameters consist of one or more numeric expressions separated by commas. A parameter has evaluated to a nonnumeric result.

**Action** Supply only numeric expressions between parentheses, and separate each expression with a comma.

**646** `The RELEASE parameter is not supported in compatibility mode.`

**Explanation** Use of the RELEASE parameter in a CONVERSE statement is not compatible with MANTIS for the IBM mainframe.

**Action** Do not use the RELEASE parameter in compatibility mode or use the Update Configuration File option in the Master User Facility to change the compatibility mode setting.

**647** `The Dollar ($) functions are not supported in compatibility mode.`

**Explanation** The dollar sign does not precede any of the reserved words in MANTIS for the IBM mainframe. MANTIS for the IBM mainframe does not recognize the $SYMBOL reserved word. There is no equivalent function in MANTIS for the IBM mainframe.

**Action** Do not use $SYMBOL in your program as this function is exclusive to MANTIS for Windows, or use the Update Configuration File option in the Master User Facility to change the compatibility mode setting.

**648** `The NEW, REPLACE, and FILE keywords are not supported in compatibility mode.`

**Explanation** The NEW, REPLACE, and FILE keywords are not supported as arguments of the ACCESS statement in MANTIS for the IBM mainframe. In MANTIS for the IBM mainframe, external files must already exist when you ACCESS them in your MANTIS program.

**Action** Create your external file before running MANTIS and do not specify the NEW, REPLACE, or FILE keyword on the ACCESS statement, or use the Update Configuration File option in the Master User Facility to change the compatibility mode setting.

**649** `Releasing an external file is not supported in compatibility mode.`

**Explanation** MANTIS for the IBM mainframe does not support closing an external file with the RELEASE *access-name* statement.

**Action** Do not use RELEASE *access-name* in your program, or use the Update Configuration File option in the Master User Facility to change the compatibility mode setting.

**650** `Indirect variable reference (&) is not supported in compatibility mode.`

**Explanation** The use of the indirect reference operator is not allowed in compatibility mode.

**Action** Refrain from using this feature, or use the Update Configuration File option in the Master User Facility to change the compatibility mode setting.

**651** `EDIT LIST is not supported in compatibility mode.`

**Explanation** The EDIT LIST statement is not supported in MANTIS for the IBM mainframe. It is not permitted in MANTIS for Windows when compatibility mode is in effect.

**Action** Refrain from using this feature, or use the Update Configuration File option in the Master User Facility to change the compatibility mode setting.

**652** `A separator is missing in the SHOW parameter list.`

**Explanation** This error can occur only when compatibility mode is selected. MANTIS for the IBM mainframe requires SHOW parameters to be separated by a comma or semicolon. MANTIS for Windows does not require a separator, allowing data to be formatted in the output buffer without intervening spaces.

**Action** Use a semicolon or a comma to separate expressions in the parameter list of the SHOW statement, or use the Update Configuration File option in the Master User Facility to change the compatibility mode setting.

**653** `A TUPLE field is larger than 64K`

**Explanation** It is not possible to a create a MANTIS file record with a TUPLE field larger than 64K. TUPLE fields contain the field definitions for SCREEN, ACCESS, and FILE profiles. The profile you are attempting to save or replace contains too many fields (more than approximately 4000).

**Action** Reduce the number of fields you have defined in your profile layout.

**658** `Warning - lines overlap, SEQUENCE issued.`

**Explanation** The program lines you copied overlap those in the existing program. MANTIS automatically issues a SEQUENCE command to renumber as many lines as necessary to restore ascending sequence.

**Action** If you relied on original line numbers, issue the LIST command and scan your program to reorient yourself.

**659** `The specified screen is not in the map set.`

**Explanation** CONVERSE *screen-name* RELEASE must specify a screen that is currently in the map set. The *screen-name* specified is not in the map set.

**Action** There are the following likely causes for this problem:

- ♦ You have not conversed the screen into the map set, or you have caused the map set to be cleared since the screen was added to the map set, either by a CONVERSE without SET/WAIT/ UPDATE, or by a CLEAR statement.
- ♦ Upon external subprogram EXIT, if any screens have been added to the map set by the subprogram, the entire map set is cleared.

**662** `Invalid environment variable name.`

**Explanation** The environment variable name used in a $SYMBOL function or SET $SYMBOL statement is invalid (e.g., because it contains characters that cannot be accepted).

**Action** Correct the environment variable name according to system requirements.

**663** `Environment variable too long.`

**Explanation** The environment variable exceeds the limit of 127 characters.

**Action** Correct the environment variable.

**664** `Value exceeds allowable range.`

**Explanation** A value in the identified line is out of range.

**Action** Check and fix the number.

**665** `External file profile key longer than external file key.`

| | |
|---|---|
| **Explanation** | The key length of the external file as specified in the external file profile is longer than the actual key length of the file. This may occur if you create a file using a profile with a partial key definition. |
| **Action** | The file must be created using an external file profile with the longest key length. |

**666** `Invalid file type in external file profile.`

| | |
|---|---|
| **Explanation** | The file type specified in the external file profile is not Indexed, Sequential, Numbered, or ASCII. |
| **Action** | Check to see if your cluster is corrupted. |

**667** `The filename is invalid.`

| | |
|---|---|
| **Explanation** | The file name specified in the ACCESS statement does not evaluate to a valid DOS file name. |
| **Action** | Check and correct the file name in the ACCESS statement. |

**668** `ACCESS, NEW, or REPLACE needs the insert password.`

| | |
|---|---|
| **Explanation** | You have specified the NEW or REPLACE parameter in an ACCESS statement, but you have used the view or alter password. |
| **Action** | Replace the invalid password with the insert password. |

**669** `The CHR function is not supported in compatibility mode.`

| | |
|---|---|
| **Explanation** | Use of the CHR function is not compatible with MANTIS for the IBM mainframe. |
| **Action** | Do not use the CHR function in compatibility mode, or use the Update Configuration File option in the Master User Facility to change the compatibility mode setting. |

**670** `The field is not entirely filled.  Please reenter.`

| | |
|---|---|
| **Explanation** | The identified field has not been filled completely. |
| **Action** | Fill out the field. |

**671** `Please reenter field with data not higher than ######.`

**Explanation** Data exceeding the specified limit has been entered.

**Action** Reenter the data to conform to the limit.

**672** `Please reenter field with data not lower than ######.`

**Explanation** Data lower than the specified limit has been entered.

**Action** Reenter the data to conform to the limit.

**673** `The field should be in MANTIS name format. Please reenter.`

**Explanation** The VALID NAME attribute was specified for this field during Screen Design, and you have entered a value that does not follow MANTIS naming conventions.

**Action** Correct the name to match MANTIS naming conventions, as documented in the *MANTIS Language* manual.

**674** `Data is invalid.  Please reenter as per mask ######.`

**Explanation** You have entered a value that does not fit the edit mask specified for the identified field during Screen Design.

**Action** Reenter the data to conform to the displayed mask.

**675** `Data is required in this field.  Please reenter.`

**Explanation** You have not entered a value for the indicated field.  The field was specified as REQUIRED during Screen Design.

**Action** Enter a value for the indicated field.

**676** `The LEVEL option of CHAIN is not supported in compatibility mode.`

**Explanation** CHAIN with LEVEL is not yet available in the IBM mainframe version of MANTIS.

**Action** You must use the Update Configuration File option in the Master User Facility to turn off compatibility mode before you can use the CHAIN ... LEVEL statement.

**680** `ERROR: Can’t find error message ###.`

**Explanation** An error has occurred, but MANTIS cannot find an error message for this error number. Either your cluster is corrupted, or the text of this message was not put into the cluster during Cincom MANTIS development.

**Action** Use the Edit MANTIS Messages option in the MASTER user to determine whether the message exists. If not, contact your Cincom representative. If the Edit Messages facility fails when trying to access the message, the cluster may be corrupted. In this case run the MANTUTIL program with the RECOVER option.

**681** `Line # is too long and has been truncated.`

**Explanation** The specified editor output line number, relative to line 1, is too long to fit into the EDIT LIST array element. The line has been truncated to the maximum length of the element.

**Action** Increase the maximum string length of the EDIT LIST variable. Alternatively, decrease the right margin in your editor.

**682** `The maximum number of lines had been exceeded at line #.`

**Explanation** There are too many output lines from the external text editor to fit into the EDIT LIST array variable. The specified line number, relative to line 1, and all subsequent output lines, are discarded by MANTIS.

**Action** Increase the dimension of the EDIT LIST array variable. The maximum allowed dimension of an array is a MANTIS option and can be adjusted using the Update Configuration File option of the MASTER User Facility.

**683** `The program logic is not correctly resolved.`

**Explanation** This is an internal MANTIS error. Please inform your Cincom representative that this error has occurred.

**Action** You can rectify this problem in one of the following ways:

- EDIT program-name
- REPLACE
- LOAD program-name
- SAVE FILE filespec
- LOAD FILE filespec
- REPLACE program-name

**684** `Breakpoints cannot be executed in a BOUND program.`

**Explanation** The current test program is BOUND. You are prevented from setting breakpoints because they will not execute when the program is run.

**Action** Use the BIND OFF command to unbind the program first.

**685** `Error in ...`

**Explanation** An error occurred in the BIND processor while parsing your program.. This message indicates what the BIND processor was parsing when the error occurred.

**Action** Correct your program and reissue the BIND command.

**686** `*** Program: # ***`

**Explanation** This message precedes any BINDing error messages with the name of the program being bound, and enables you to correlate the error messages with programs when multiple programs are being bound.

**Action** None. Informational only.

**687** `The bound program will not run in compatibility mode.`

**Explanation** IBM compatibility mode is currently in effect, and the BOUND program contains syntax that is not IBM compatible. If the program was BOUND in IBM compatibility mode, then the relevant diagnostic warning messages would have been displayed at that time.

**Action** The program cannot be executed while IBM compatibility mode is in effect.

**688** `Value not in fixed list.  Use VALIDINFO for values.`

**Explanation** The value in the field is not in the fixed list of allowed values for the field. This list was specified when the screen was designed.

**Action** Please use the VALIDINFO function key (normally CTRL-V) to inspect the values allowed for the field.

**689** `Value not in named variable.  Use VALIDINFO for values.`

**Explanation** The value in the field is not one of the values in the named variable for the field. This variable name was specified when the screen was designed and is usually used as an array of correct values.

**Action** Please use the VALIDINFO function key (normally CTRL-V) to inspect the values allowed for the field.

**690**

```
Value not in named variable or fixed list.  Use VALIDINFO
for values.
```

**Explanation** The value in the field is not in the fixed list of allowed values for the field or in the list of values contained in the named variable. This list and name were specified when the screen was designed.

**Action** Please use the VALIDINFO function key (normally CTRL-V) to inspect the values currently allowed for the field.

**691** `Named variable (####) wrong type.`

**Explanation** During the processing of a SCREEN statement, a field that has the extended editing MANTIS named variable was found. This variable already exists, but is the wrong type.

**Action** Correct the variable type.

**692** `Named array (####) has too many dimensions.`

**Explanation** During the processing of a SCREEN statement, a field that has the extended editing MANTIS named variable was found. This variable already exists, is the correct type, and is an array but has too many dimensions (greater than one).

**Action** Correct the number of dimensions in the named array.

**697** `The number of CHAIN arguments must be exact in compatibility mode.`

**Explanation** The number of arguments in the CHAIN statement must be the same as the number of formal arguments in the ENTRY statement of the CHAINed-to program.

**Action** Match the argument lists in the CHAIN and ENTRY statements.

**698** `All parameters must be defined in the external DO in compatibility mode.`

**Explanation** One or more of the parameters to the DO statement have not been defined prior to the DO statement. Note that the DO statement executes an external MANTIS program. In IBM compatibility mode, all parameters to external DO statements must be defined.

**Action** Define all parameters prior to the DO statement, or switch off IBM compatibility mode.

**700** `The MANTIS FILE is obsolete and requires conversion.`

**Explanation** The MANTIS library/directory/cluster predates the current version of MANTIS. Either the whole MANTIS file is out of date, or the entity being accessed is out of date. If this error occurs during sign-on to MANTIS, the whole MANTIS file is obsolete and has not been converted during installation of the new release of MANTIS, as required. If the MANTIS file is correct, but the particular entity being accessed is causing the problem, an old-format entity must have been inserted into the MANTIS file after conversion; the most likely vehicle for this is the Transfer Facility.

**Action** Check that the installation procedure was followed correctly and that your MANTIS file has been converted for the current MANTIS release.

**703** `The BOUND program contains an invalid operation code or is corrupt.`

**Explanation** The BOUND program interpreter has trapped an invalid operation code. Either the BIND command generated bad code, or the BOUND program profile is corrupt.

**Action** Please report this error to your CINCOM representative.

**705** `The Pcode Interpreter stack has overflowed (development only).`

**Explanation** The BOUND program interpreter has exhausted its internal evaluation stack. This is probably caused by an extremely complex expression.

**Action** Simplify the expression.

**706** `The PCI heap is too small for a text expression to be evaluated.`

**Explanation** MANTIS uses a heap during the evaluation of text expressions. A text value in the current expression will not fit on the heap.

**Action** The size of the heap is a MANTIS option and can be adjusted by using the Update Configuration File option by the Master User.

**801** `An internal binding error has occurred (stack underflow)`

**Explanation** The parse of the MANTIS program has failed due to an internal MANTIS error.

**Action** Report this error to your CINCOM representative.

**802** The Binding operation has been terminated.

**Explanation** Syntax errors in the MANTIS program are such that the binding operation cannot continue to the end of the program. Normally, program syntax errors are recoverable, allowing the whole program to be parsed.

**Action** This is an informational message only, and does not require any corrective action**.**

**803** `The MANTIS language parse table has an invalid entry.`

**Explanation** An unexpected condition has revealed an error in the MANTIS parse table.

**Action** Please report this error to your CINCOM representative.

**804** `Too many syntax errors, the binding operation has been terminated.`

**Explanation** The number of errors encountered during the BIND operation exceeds the maximum allowed number of errors.

**Action** The maximum number of BIND errors allowed is a MANTIS option that can be set using the Update Configuration File option of the Master User Facility.

**805** `An internal SQL binding error has occurred (missing hostvar).`

**Explanation** An internal MANTIS error has occurred while BINDing an SQL statement.

**Action** Contact your Cincom representative.

**901** `A parameter is missing or incorrectly specified.`

**Explanation** One of the parameters on an INTERNAL statement is missing or is incorrectly specified. INTERNAL statements are reserved for use by Cincom only.

**Action** Contact your Cincom representative.

**903** `The usercode parameter is invalid.`

**Explanation** The usercode parameter must be numeric and must not exceed the maximum user code allowed.

**Action** Contact your Cincom representative.

**905** `The function parameter is invalid.`

**Explanation** The function parameter must be a text expression that evaluates to either *R* or *W*.

**Action** Contact your Cincom representative.

**908** `The INTERNAL number is not defined.`

**Explanation** The first parameter on an INTERNAL statement is the internal number. The specified number has not been defined to MANTIS.

**Action** Contact your Cincom representative.

**909** `You are not authorized to execute the INTERNAL statement.`

**Explanation** The INTERNAL statement is reserved for Cincom use only.

**Action** Refrain from using the INTERNAL statement.

**910** `The relative field number parameter is too large.`

**Explanation** The relative field number parameter must refer to an existing field or be one greater than the number of fields when adding new fields.

**Action** Contact your Cincom representative.

| | | |
|---|---|---|
| **911** | `A text parameter is the wrong length.` | |
| | **Explanation** | Either the maximum length of the text parameter or the number of dimensions is too small. |
| | **Action** | Contact your Cincom representative. |
| **912** | `The TRANSFER file could not be opened.` | |
| | **Explanation** | MANTIS received an error status when attempting to open the Transfer file. |
| | **Action** | Obtain diagnostic information by going into Program Design and using the HELP LAST command to display the last message. |
| **913** | `The TRANSFER file is not open.` | |
| | **Explanation** | The Transfer file should be open when this statement is executed. |
| | **Action** | Contact your Cincom representative. |
| **914** | `The screen is too large.` | |
| | **Explanation** | You have tried to create a screen that requires memory in excess of 64K. |
| | **Action** | Simplify the screen design (e.g., reduce its dimensions, remove fields or attributes). If necessary, break the screen into multiple maps. |
| **916** | `The TRANSFER file is incompatible with this release.` | |
| | **Explanation** | MANTIS cannot use the TRANSFER file because it is not compatible with this release of MANTIS. |
| | **Action** | Upgrade the TRANSFER file before using it. |
| **926** | `The screen field number is invalid.` | |
| | **Explanation** | This is an internal error. |
| | **Action** | Contact your Cincom representative. |

**931** `Attempting to delete an in-use file.`

**Explanation** You have used the REPLACE parameter on an ACCESS statement, but the external file that you are attempting to replace has already been opened by another ACCESS statement. This can be caused by ACCESS statements in the wrong order, or by specifying Delayed Create in the external file view design for a file that you are using with more than one ACCESS.

**Action** Either change the Delayed Create option in the file view to N, or alter the order of your ACCESS statements.

**932** `The INTERFACE statement is not supported in this release of MANTIS.`

**Explanation** You have tried to execute an INTERFACE statement, but this version of MANTIS does not support this statement.

**Action** Change your program, or upgrade to a version of MANTIS with interface support.

**933** `Inconsistent ACCESS views.`

**Explanation** You have tried to access an external file with two different views, where one view defines the file as indexed and the other does not.

**Action** Correct your views or your program.

**934** `The CALL statement is not supported in this release of MANTIS.`

**Explanation** You have tried to execute a CALL statement, but this version of MANTIS does not support this statement.

**Action** Change your program, or upgrade to a version of MANTIS with interface support.

**943** `The MARK statement is not supported in this elease of MANTIS.`

**Explanation** You have tried to execute a MARK statement, but this version of MANTIS does not support this statement.

**Action** Change your program.

**950** `Security violation - security block missing or invalid.`

| | |
|---|---|
| **Explanation** | The security block has been removed from the printer port. |
| **Action** | Replace the security block immediately. |

# Index

## $

$SYMBOL function 218, 296

## A

ABS function 74
ACCESS statement 75, 290
arithmetic
  expressions 58
  logical expressions 60
  logical operators 61
  operators 58
  relational expressions 60
  relational operands 61
  unary operator 59
arrays 45
ASCII 288
at sign (@) 41
ATN function 80
ATTRIBUTE statement 81, 290
ATTRIBUTE(PRINTER) function 87
automatic mapping 276

## B

BIG statement 45, 55, 88
BREAK statement 89, 291
building maps 252
built-in functions 64, 65

## C

CALL statement 90
CHAIN statement 93, 271, 291
character set 40
CHR function 95, 291
CLEAR statement 96, 291
clearing
  map 264
  map set 265
collating sequence
  ASCII 288
  EBCDIC 288
command
  DOWN 267
  UP 267
commands 47
comments 49, 286
COMMIT statement 98, 291
compatibility mode 20, 284
CONVERSE statement 100, 252, 254, 292
COS function 107
CTRL-BREAK 19, 28
CURSOR function 108

## D

DATAFREE function 110
DATE function 111
DATE statement 112
debugging 278
DELETE statement 114
DEQUEUE statement 118
display modes 21
dissimilarity debugging 297
DO statement 119, 267, 271, 292
DOLEVEL 267, 268
DOLEVEL function 122
dollar sign ($) 41
DOWN command 267

## E

E function 123
EBCDIC 288
EDIT statement 124, 292
editing keys 32
ENQUEUE statement 125
ENTRY-EXIT statement 126
error messages. *see* messages
EXP function 128
external DO. *see* DO statement

## F

facility selection menu 38
FALSE function 129
field input 288
FILE statement 130

floating maps 22, 26
FOR-END statement 134, 292
FORMAT function 136
frequency of DO statements 278
FSI function 137, 293
full-screen mode
  fields 24
  using 22
function
  $SYMBOL 218, 296
  ABS 74
  ATN 80
  ATTRIBUTE(PRINTER) 87
  CHR 95, 291
  COS 107
  CURSOR 108
  DATAFREE 110
  DATE 111
  DOLEVEL 122
  E 123
  EXP 128
  FALSE 129
  FORMAT 136
  FSI 137, 293
  INT 155
  KEY 159, 293
  LANGUAGE 160, 293
  LOG 164
  MODIFIED 165
  NOT 168
  NULL 168, 294
  NUMERIC 169, 294
  PASSWORD 177
  PI 180
  POINT 181
  PRINTER 182
  PROGFREE 186
  RELEASE 193
  RND 196
  SGN 205
  SIN 209
  SIZE 210
  SQR 216
  TAN 219
  TERMINAL 220
  TERMSIZE 221
  TIME 224
  TRUE 231
  TXT 232
  UPPERCASE 241, 296
  USER 242
  USERWORDS 243
  VALUE 244
  ZERO 250
functions, built-in 64, 65

G

GET statement 139, 293

H

HEAD statement 147

I

IF-ELSE-END statement 149
indirect reference 41, 63, 286
INSERT statement 151
INT function 155
INTERFACE statement 156

K

Kanji support 286
key assignments 286
KEY function 159, 293
key simulation field 23, 24, 27, 28, 31, 36
keyboard operation 30
keyboard, using the 32
keys
  editing 32
  windowing 32
keys. *see* logical keys
KILL 28, 31

L

language conventions 39
LANGUAGE function 160, 293
LET statement 161
LOG function 164
logical
  display 21, 29
  keys 19, 20
  terminal interface 251
logical unit of work 71

**M**

message field 23, 24
messages 299
MODIFIED function 165
modularizing 275
multiple images of single screen
    design 259

**N**

NEXT statement 167, 293
noninteractive input 288
NOT function 168
NULL function 168, 294
numeric
  considerations 54
  scientific notation 54
  storing data 54
NUMERIC function 169, 294

**O**

OBTAIN statement 170
operands, text 53
operators
  arithmetic 58
  logical 61
  unary 59
OUTPUT statement 172, 294

**P**

PAD statement 174
passing arguments 268
PASSWORD function 177
PERFORM statement 178, 294
performance 280
personal computer
    considerations 19
PI function 180
POINT function 181
PRINTER function 182
PRINTER= statement 183, 294
PROGFREE function 186
program architecture 269
program lines 48
PROGRAM statement 187
programming hints 281
PROMPT statement 189

**R**

relational operands 61
RELEASE function 193
RELEASE statement 190, 295
reserved words 42
RESET statement 194
RETURN statement 195, 295
RND function 196
row/column coordinates 22, 24,
    27

**S**

scientific notation 54
screen design 287
SCREEN statement 197
scroll map 25
scroll mode 25
SCROLL statement 199, 295
scroll-mode fields 26
SEED statement 201
SET statement 202, 295
SGN function 205
SHOW statement 206, 295
Signing on to MANTIS 37
SIN function 209
SIZE function 210
size limits 285
SLICE statement 212, 296
SLOT statement 214, 296
SMALL statement 55, 215
special characters 40
SQR function 216
statement
  ACCESS 75, 290
  ATTRIBUTE 81, 290
  BIG 88
  BREAK 89, 291
  CALL 90
  CHAIN 93, 271, 291
  CLEAR 96, 291
  COMMIT 98, 291
  CONVERSE 100, 252, 254,
    292
  DATE 112
  DELETE 114

statement (*cont.*)
  DEQUEUE 118
  DO 119, 267, 271, 292
  EDIT 124, 292
  ENQUEUE 125
  ENTRY-EXIT 126
  FILE 130
  FOR-END 134, 292
  GET 139, 293
  HEAD 147
  IF-ELSE-END 149
  INSERT 151
  INTERFACE 156
  LET 161
  NEXT 167, 293
  OBTAIN 170
  OUTPUT 172, 294
  PAD 174
  PERFORM 178, 294
  PRINTER= 183, 294
  PROGRAM 187
  PROMPT 189
  RELEASE 190, 295
  RESET 194
  RETURN 195, 295
  SCREEN 197
  SCROLL 199, 295
  SEED 201
  SET 202, 295
  SHOW 206, 295
  SLICE 212, 296
  SLOT 214, 296
  SMALL 215
  STOP 217
  TEXT 222
  TIME 226
  TRAP 228
  UNPAD 233
  UNTIL-END 236
  UPDATE 237
  WAIT 245
  WHEN-END 246
  WHILE-END 248
statements 47
STOP statement 217
storing data
  numeric 54
  text 50
symbolic names 44

**T**

TAN function 219
TERMINAL function 220
terminology 20
TERMSIZE function 221
text
  considerations 50
  expressions 53
  operands 53
  storing data 50
  substrings 51
TEXT statement 222
TIME function 224
TIME statement 226
TRAP statement 228
TRUE function 231
TXT function 232

**U**

unary operator 59
UNPAD statement 233
unsolicited input field 24, 26, 27
UNTIL-END statement 236
UP command 267
UPDATE statement 237
UPPERCASE function 241, 296
USER function 242
USERWORDS function 243
Using the keyboard 30

**V**

VALUE function 244
variables 45

**W**

WAIT statement 245
WHEN-END statement 246
WHILE-END statement 248
windowing 251, 260
windowing keys 32

**Z**

ZERO function 250

# Reader Comment Sheet

Name: ______________________________

Job title/function: ______________________________

Company name: ______________________________

Address: ______________________________

Telephone number: ____________________ Date: ____________

How often do you use this manual? ☐ Daily ☐ Weekly ☐ Monthly ☐ Less

How long have you been using this product? ☐ Months ☐ Years

Can you find the information you need? ☐ Yes ☐ No Please comment.

______________________________

______________________________

Is the information easy to understand? ☐ Yes ☐ No Please comment.

______________________________

______________________________

Is the information adequate to perform your task? ☐ Yes ☐ No Please comment.

______________________________

______________________________

General comment: ______________________________

______________________________

______________________________

______________________________

______________________________

______________________________

To respond, please fax to Larry Fasse at (513) 612-2000.

CINCOM The Smart Choice